信创大赛参考资料-基础篇

武汉达梦数据库有限公司内部资料

达梦数据库信创大赛资料

目 录

第 1 章 DM 安装及卸载 1
1.1 Linux（Unix）下 DM 的安装与卸载 1
1.1.1 安装前准备工作 1
1.1.2 安装 DM 3
1.1.3 卸载 DM 14
第 2 章 创建和配置 DM 数据库 18
2.1 创建 DM 数据库 18
2.2 使用数据库配置工具创建数据库 19
2.2.1 启动数据库配置助手 19
2.2.2 使用数据库配置工具创建数据库 20
2.3 使用 dminit 创建数据库 30
2.4 注册数据库服务 34
2.5 查看数据库信息 36
2.6 删除数据库 38
2.7 删除数据库服务 41
第 3 章 启动和关闭数据库 45
3.1 启动数据库 45
3.1.1 Linux 系统 45
3.1.2 检查 LICENSE 45
3.2 数据库状态和模式 45
3.3 关闭数据库 46
3.3.1 Linux 系统 46
第 4 章 DM 逻辑结构概述 47
4.1 数据库和实例 47
4.1.1 数据库 47
4.1.2 实例 47
4.2 DM 逻辑存储结构 48
4.2.1 表空间 48
4.2.2 记录 49
4.2.3 页 49
4.2.4 簇 50
4.2.5 段 51
第 5 章 DM 物理存储结构 53
5.1 配置文件 53
5.1.1 DM 数据库服务配置 54
5.2 控制文件 54
5.3 数据文件 55
5.4 重做日志文件 55
5.5 归档日志文件 56
5.6 备份文件 56
第 6 章 管理表 56
6.1 管理表的准则 57
6.1.1 设计表 57
6.1.2 指定表的存储空间上限 57

达梦数据库信创大赛资料

6.1.3 指定表的存储位置......58
6.2 创建表......58
6.2.1 创建普通表......58
6.2.2 指定表的聚集索引......58
6.2.3 查询建表......59
6.3 更改表......59
6.4 删除表......60
6.5 清空表......60
6.5.1 使用 DELETE......60
6.5.2 使用 DROP 和 CREATE......60
6.5.3 使用 TRUNCATE......61
6.6 查看表信息......61
6.6.1 查看表定义......61
6.6.2 查看表的空间使用情况......61
第 7 章 管理索引......62
7.1 创建索引......62
7.1.1 明确地创建索引......62
7.1.2 创建聚集索引......62
7.1.3 明确地创建唯一索引......63
7.1.4 自动创建与约束相关的唯一索引......63
7.2 重建索引......63
7.3 删除索引......63
7.4 查看索引信息......64
第 8 章 管理视图......65
8.1 管理视图......65
8.2 查看视图......65
第 9 章 结构化查询语言 DM_SQL 简介......67
9.1 DM_SQL 语言的特点......67
1．功能一体化......67
2．两种用户接口使用统一语法结构的语言......68
3．高度非过程化......68
4．面向集合的操作方式......68
5．语言简洁，方便易学......68
9.2 保留字与标识符......68
9.3 DM_SQL 语言的功能及语句......68
9.4 DM_SQL 所支持的数据类型......69
9.4.1 常规数据类型......69
9.4.2 日期时间数据类型......70
9.5 DM_SQL 语言支持的表达式......71
9.5.1 字符串表达式......71
9.5.2 时间值表达式......71
第 10 章 数据查询语句......72
10.1 单表查询......74
10.1.1 简单查询......75
10.1.2 带条件查询......76
10.1.3 集函数......79
10.2 连接查询......80

达梦数据库信创大赛资料

10.2.1 JOIN … ON......81
10.2.2 外连接(OUTER JOIN)......81
10.3 子查询......86
10.3.1 表子查询......86
10.4 合并查询结果......89
10.5 分组查询......91
10.5.1 GROUP BY 子句的使用......91
10.5.2 HAVING 子句的使用......92
10.6 ORDER BY 子句......93
第 11 章 数据的插入、删除和修改......94
11.1 数据插入语句......94
11.2 数据修改语句......97
11.3 数据删除语句......98
第 12 章 自主访问控制......99
12.1 权限管理......99
12.1.1 数据库权限......99
12.1.2 对象权限......100
12.2 角色管理......101
12.2.1 角色的创建与删除......102
12.2.2 角色的启用与禁用......105
12.3 权限的分配与回收......105
12.3.1 数据库权限的分配与回收......105
12.3.2 对象权限的分配与回收......107
12.3.3 角色权限的分配与回收......110

达梦数据库信创大赛资料

第 1 章　DM 安装及卸载

DM 在各个操作系统下的数据库服务器版本具有相同的内核，本章介绍 DM 在 Windows、Linux(Unix)操作系统下的安装。

注：根据版本的用途，安装 DM 程序后，默认装有一个许可证(License)。如果用户想拥有更多授权的许可证，请向达梦公司申请或购买。

1.1 Linux（Unix）下 DM 的安装与卸载

1.1.1 安装前准备工作

用户在安装 DM 之前需要检查或修改操作系统的配置，以保证 DM 正确安装和运行。

标题及下文中提到的 Linux(Unix)，包括 Linux、AIX、HP-UNIX、Solaris 和 FreeBSD 操作系统。以下安装程序说明将以 Red Hat Enterprise Linux 6 for x86-64 系统为例，由于不同操作系统系统命令不尽相同，具体步骤及操作请以本机系统为准，具体细节可向系统管理员咨询。

1)　检查 Linux(Unix)系统信息

如果用户的 DM 软件安装包是经过数字签名的，请参照以下步骤进行操作，其他情况请忽略此步骤。

软件安装包的数字签名校验：

1. 导入达梦公司的公钥，命令如下：
   #gpg --import dm-pub-key
2. 将达梦公司的公钥添加到绝对信任列表
   #gpg --edit-key 武汉达梦数据库有限公司 trust
3. 验证达梦安装包的数字签名
   # gpg --verify dm.sign dm8_setup_xxx.iso

步骤 3 输出结果是“完好的签名”(“Good Signature”)则表示安装包文件完好无损。

用户在安装 DM 前，需要检查当前操作系统的相关信息，确认 DM 安装程序与当前操作系统匹配，以保证 DM 能够正确安装和运行。用户可以使用以下命令检查操作系统基本信息。如下图所示：

```
#获取系统位数
getconf LONG_BIT
#查询操作系统 release 信息
lsb_release -a
#查询系统信息
cat /etc/issue
#查询系统名称
uname -a
```

达梦数据库信创大赛资料

```
[root@localhost ~]# getconf LONG_BIT
64
[root@localhost ~]# lsb_release -a
LSB Version:    :core-4.0-amd64:core-4.0-noarch:graphics-4.0-amd64:graphics-4.0-noarch:printing-4.0-amd64:printing-4.0-noarch
Distributor ID: RedHatEnterpriseServer
Description:    Red Hat Enterprise Linux Server release 6.2 (Santiago)
Release:        6.2
Codename:       Santiago
[root@localhost ~]# cat /etc/issue
Red Hat Enterprise Linux Server release 6.2 (Santiago)
Kernel \r on an \m

[root@localhost ~]# uname -a
Linux localhost.localdomain 2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64 GNU/Linux
[root@localhost ~]#
```

图 1.20 检查 Linux(Unix)系统信息

2） 创建安装用户

为了减少对操作系统的影响，用户不应该以 root 系统用户来安装和运行 DM。用户可以在安装之前为 DM 创建一个专用的系统用户。可参考以下示例创建系统用户和组(并指定用户 ID 和组 ID)，具体细节可向系统管理员咨询。

1. 创建安装用户组 dinstall。

```
groupadd -g 12349 dinstall
```

2. 创建安装用户 dmdba。

```
useradd -u 12345 -g dinstall -m -d /home/dmdba -s /bin/bash dmdba
```

3. 初始化用户密码。

```
passwd dmdba
```

4. 之后通过系统提示进行密码设置。

注意：创建安装系统用户完成后，安装手册之后的操作默认使用安装系统用户进行操作。

3） Linux(Unix)下检查操作系统限制

在 Linux(Unix)系统中，因为 ulimit 命令的存在，会对程序使用操作系统资源进行限制。为了使 DM 能够正常运行，建议用户检查当前安装用户的 ulimit 参数。

运行 ulimit -a 进行查询。如下图所示：

```
[root@localhost ~]# ulimit -a
core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 2066416
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 1024
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
[root@localhost ~]#
```

达梦数据库信创大赛资料

图 1.21 查看 ulimit 参数

参数使用限制：

1.data seg size

```
data seg size (kbytes, -d)
```

建议用户设置为 1048576(即 1GB)以上或 unlimited(无限制)，此参数过小将导致数据库启动失败。

2. file size

```
file size(blocks, -f)
```

建议用户设置为 unlimited(无限制)，此参数过小将导致数据库安装或初始化失败。

3. open files

```
open files(-n)
```

建议用户设置为 65536 以上或 unlimited(无限制)。

4.virtual memory

```
virtual memory (kbytes, -v)
```

建议用户设置为 1048576(即 1GB)以上或 unlimited(无限制)，此参数过小将导致数据库启动失败。

如果用户需要为当前安装用户更改 ulimit 的资源限制，请修改文件 /etc/security/limits.conf。

4) 设置 JAVA 环境

如果对安装程序和客户端运行的 JAVA 环境无特殊需求，可忽略此步骤。在运行安装程序之前，通过设置环境变量 DM_JAVA_HOME 指定特定的 JAVA 的目录，安装程序将使用此 JAVA 环境进行安装，且安装完成之后，客户端程序也将使用此 JAVA 环境。具体命令如下所示：

```
#以 BASH 为例：
DM_JAVA_HOME=/mount_point/jdk_home_dir
export DM_JAVA_HOME
```

1.1.2 安装 DM

用户应登录或切换到安装系统用户，进行以下安装步骤的操作(注：不建议使用 root 系统用户进行安装)。

将 DM 安装光盘放入光驱，然后加载(mount)光驱。一般可以通过执行下面的命令来加载光驱：

```
mount /dev/cdrom /mnt/cdrom
```

这里假定光驱对应的文件为/dev/cdrom 且目标路径/mnt/cdrom 已存在。

加载(mount)光驱后，在/mnt/cdrom 目录下存在 DMInstall.bin 文件，DMInstall.bin 文件就是 DM 的安装程序。在运行安装程序前，需要赋予 DMInstall.bin 文件执行权限。具体命令如下所示：

```
chmod 755 ./DMInstall.bin
```

达梦数据库信创大赛资料

1） 图形化安装

用户双击 DMInstall.bin 或执行以下命令将运行 DM 的图形化安装。

```
./DMInstall.bin
```

安装须知：

1） 用户在进行图形化安装时，应当确认当前正处于图形化界面的运行环境，否则运行安装程序将报错。这种情况建议用户使用命令行安装 DM。

2） 建议用户使用安装系统用户直接登录。如果用户在图形化界面中使用 su 命令切换至安装系统用户，可能导致启动图形化安装程序启动失败。具体情况请参考附录 1：xhost 配置。

安装过程如下：

步骤 1：提示对话框

如果当前操作系统中已存在 DM，将弹出提示对话框。如下图所示：

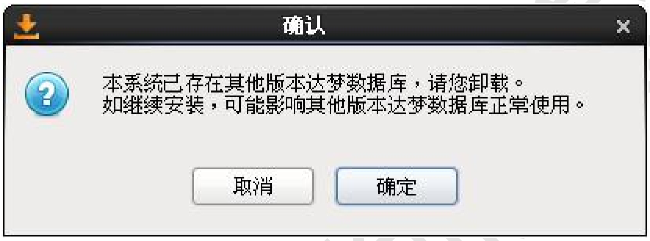


图 1.24 确认

注意：若系统中已安装 DM，重新安装前，应完全卸载原来的 DM。并且在重新安装前，务必备份好数据。

步骤 2：选择语言和时区

请根据系统配置选择相应语言与时区，点击“确定”按钮继续安装。如下图所示：



图 1.25 选择语言与时区

步骤 3：欢迎页面

点击“开始”按钮继续安装。如下图所示：



图 1.26 欢迎界面

步骤 4：许可证协议

在安装和使用 **DM** 之前，该安装程序需要用户阅读许可协议条款，用户如接受该协议，则选中“接受”，并点击“下一步”继续安装；用户若选中“不接受”，将无法进行安装。如下图所示：



图 1.27 许可证协议

步骤 5：查看版本信息

用户可以查看 DM 服务器、客户端等各组件相应的版本信息。如下图所示：



图 1.28 版本信息

步骤 6：验证 Key 文件

用户点击“浏览”按钮，选取 Key 文件，安装程序将自动验证 Key 文件信息。如果是合法的 Key 文件且在有效期内，用户可以点击"下一步"继续安装。如下图所示：



图 1.29 Key 文件

步骤 7：选择安装方式

DM 安装程序提供四种安装方式：“典型安装”、“服务器安装”、“客户端安装”和“自定义安装”，用户可根据实际情况灵活地选择。如下图所示：

典型安装包括：服务器、客户端、驱动、用户手册、数据库服务。

服务器安装包括：服务器、驱动、用户手册、数据库服务。

客户端安装包括：客户端、驱动、用户手册。

自定义安装包括：根据用户需要勾选组件，可以是服务器、客户端、驱动、用户手册、数据库服务中的任意组合。

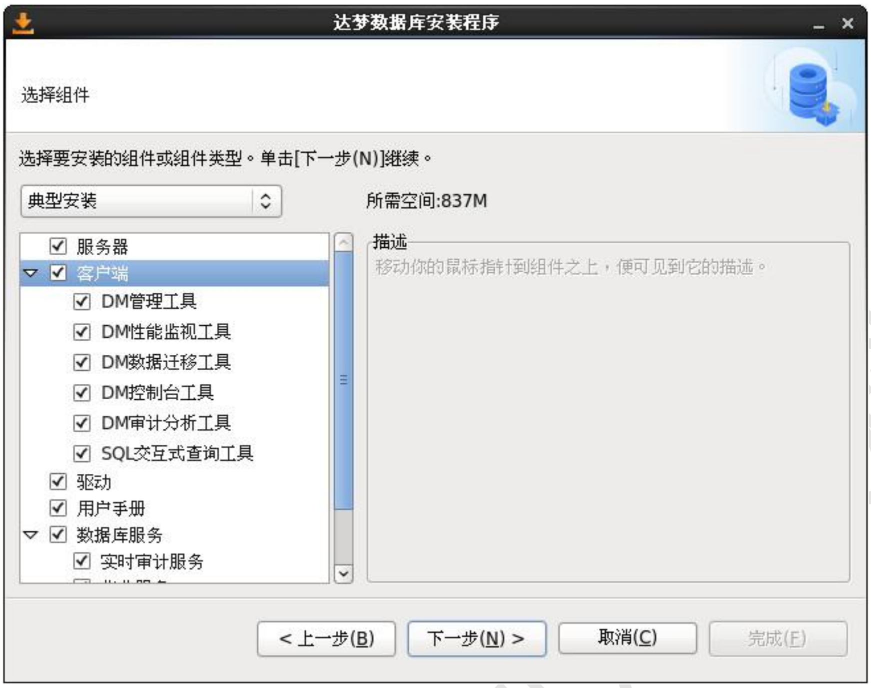


图 1.30 选择组件

一般地，作为服务器端的机器只需选择“服务器安装”选项，特殊情况下，服务器端的机器也可以作为客户机使用，这时，机器必须安装相应的客户端软件。

步骤 8：选择安装目录

选择安装目录。如下图所示。



图 1.31 选择安装位置

DM 默认安装目录为$HOME/dmdbms（如果安装用户为 root 系统用户，则默认安装目录为/opt/dmdbms，但不建议使用 root 系统用户来安装 DM），用户可以通过点击“浏览”按钮自定义安装目录。如果用户所指定的目录已经存在，则弹出如下图所示警告消息框提示

用户该路径已经存在。若确定在指定路径下安装请点击“确定”，则该路径下已经存在的 DM 某些组件，将会被覆盖；否则点击“取消”，返回到上图所示界面，重新选择安装目录。

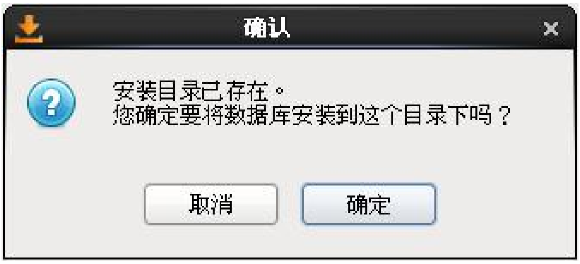


图 1.32 确认安装目录

说明：安装路径里的目录名由英文字母、数字和下划线等组成，不建议使用包含空格和中文字符的路径等。

步骤 9：安装前小结

显示用户即将进行的安装的有关信息，例如产品名称、版本信息、安装类型、安装目录、可用空间、可用内存等信息，用户检查无误后点击“安装”按钮，开始拷贝安装的软件。如下图所示：



图 1.33 安装前小结信息

步骤 10：安装

安装。如下图所示。



图 1.34 安装界面

注意：当安装进度完成时将会弹出对话框，提示使用 **root** 系统用户执行相关命令。用户可根据对话框的说明完成相关操作，之后可关闭此对话框，点击“完成”按钮结束安装。如下图所示：

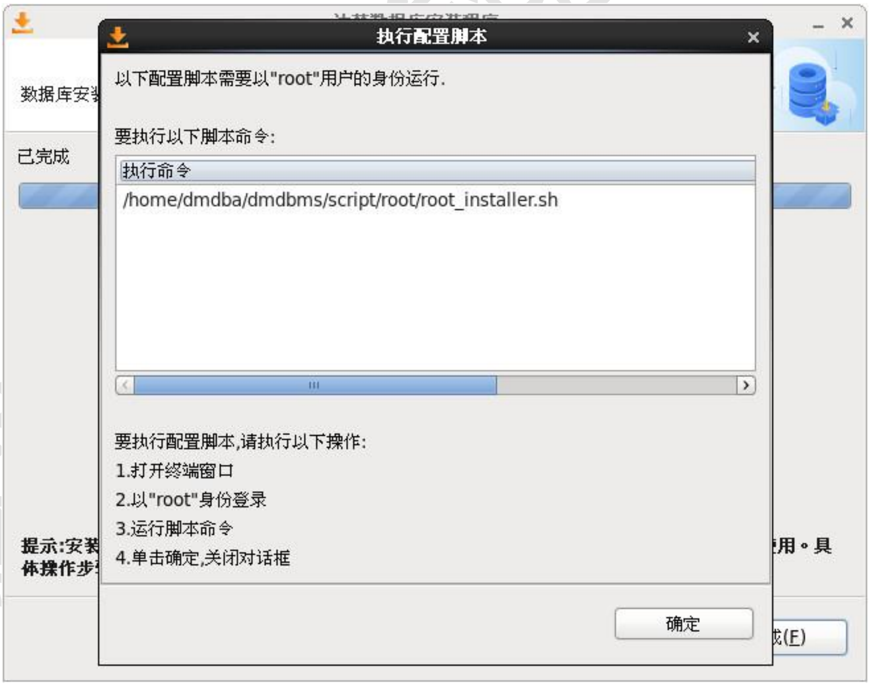


图 1.35 执行配置脚本

步骤 11：初始化数据库

如用户在选择安装组件时选中服务器组件，DM 安装过程结束时，将会提示是否初始化数据库，如下图所示。若用户未安装服务器组件，安装完成后，点击“完成”将直接退出，点击“取消”将完成安装，关闭对话框。



图 1.36 初始化数据库

若用户选中创建数据库选项，点击“初始化”将弹出数据库配置工具。如下图所示：



图 1.37 达梦数据库配置助手

2） 命令行安装

在现实中，许多 Linux(Unix)操作系统上是没有图形化界面的，为了使 DM 能够在这些操作系统上顺利安装，DM 提供了命令行的安装方式。在终端进入到安装程序所在文件夹，执行以下命令进行命令行安装：

```
./DMInstall.bin -i
```

安装过程如下：

步骤 1：选择安装语言

请根据系统配置选择相应语言，输入选项，回车进行下一步。如下图所示：

```
[dmdba@localhost ~]$ ./DMInstall.bin -i
请选择安装语言(C/c:中文 E/e:英文) [C/c]:
```

图 1.38 选择安装语言

如果当前操作系统中已存在 DM，将在终端弹出提示，输入选项：继续，将进行下一步的命令行安装，否则退出命令行安装。如下图所示：

```
本系统已存在其他版本达梦数据库，请您卸载。如继续安装，可能影响其他版本达梦数据库正常使用。
是否继续? (Y/y:是 N/n:否) [Y/y]:
```

图 1.39 是否继续

注意：若操作系统中已安装 DM，重新安装前，应完全卸载已存在的 DM。并且在重新安装前，务必备份好数据。

步骤 2：验证 Key 文件

用户可以选择是否输入 Key 文件路径。不输入则进入下一步安装，输入 Key 文件路径，安装程序将显示 Key 文件的详细信息，如果是合法的 Key 文件且在有效期内，用户可以继续安装。如下图所示：

```
欢迎使用达梦数据库安装程序

是否输入Key文件路径? (Y/y:是 N/n:否) [Y/y]:y
请输入Key文件的路径地址 [dm.key]:/home/dmdba/dm.key

有效日期: 无限制
服务器颁布类型: 企业版
发布类型: 测试版
用户名称: 测试用户
授权用户数: 无限制
并发连接数: 无限制
```

图 1.40 验证 Key 文件

步骤 3：输入时区

用户可以选择 DM 的时区信息。如下图所示：

```
是否设置时区? (Y/y:是 N/n:否) [Y/y]:y
设置时区:
[ 1]: GTM-12=日界线西
[ 2]: GTM-11=萨摩亚群岛
[ 3]: GTM-10=夏威夷
[ 4]: GTM-09=阿拉斯加
[ 5]: GTM-08=太平洋时间（美国和加拿大）
[ 6]: GTM-07=亚利桑那
[ 7]: GTM-06=中部时间（美国和加拿大）
[ 8]: GTM-05=东部部时间（美国和加拿大）
[ 9]: GTM-04=大西洋时间（美国和加拿大）
[10]: GTM-03=巴西利亚
[11]: GTM-02=中大西洋
[12]: GTM-01=亚速尔群岛
[13]: GTM=格林威治标准时间
[14]: GTM+01=萨拉热窝
[15]: GTM+02=开罗
[16]: GTM+03=莫斯科
[17]: GTM+04=阿布扎比
[18]: GTM+05=伊斯兰堡
[19]: GTM+06=达卡
[20]: GTM+07=曼谷，河内
[21]: GTM+08=中国标准时间
[22]: GTM+09=汉城
[23]: GTM+10=关岛
[24]: GTM+11=所罗门群岛
[25]: GTM+12=斐济
[26]: GTM+13=努库阿勒法
[27]: GTM+14=基里巴斯
请选择设置时区 [21]:21
```

图 1.41 输入时区

步骤 4：选择安装类型

命令行安装与图形化安装的选择的安装类型是一样。如下图所示：

```
安装类型:
1 典型安装
2 服务器
3 客户端
4 自定义
请选择安装类型的数字序号 [1 典型安装]:4
1 服务器组件
2 客户端组件
  2.1 DM管理工具
  2.2 DM性能监视工具
  2.3 DM数据迁移工具
  2.4 DM控制台工具
  2.5 DM审计分析工具
  2.6 SQL交互式查询工具
3 驱动
  3.1 ODBC驱动
  3.2 JDBC驱动
4 用户手册
5 数据库服务
  5.1 实时审计服务
  5.2 作业服务
  5.3 实例监控服务
  5.4 辅助插件服务
请选择安装组件的序号 (使用空格间隔) [1 2 3 4 5]:1 2 3 4 5
所需空间: 733M
```

图 1.42 选择安装类型

用户选择安装类型需要手动输入，默认是典型安装。如果用户选择自定义安装，将打印

全部安装组件信息。用户通过命令行窗口输入要安装的组件序号，选择多个安装组件时需要使用空格进行间隔。输入完需要安装的组件序号后回车，将打印安装选择组件所需要的存储空间大小。

步骤 4：选择安装路径

用户可以输入 DM 的安装路径，不输入则使用默认路径，默认值为$HOME/dmdbms(如果安装用户为 root，则默认安装目录为/opt/dmdbms，但不建议使用 root 系统用户来安装 DM)。　　如下图所示：

```
请选择安装目录 [/home/dmdba/dmdbms]:/home/dmdba/dmdbms
可用空间: 7963M
是否确认安装路径?(Y/y:是 N/n:否)  [Y/y]:y
```

图 1.43 选择安装路径

安装程序将打印当前安装路径的可用空间，如果空间不足，用户需重新选择安装路径。如果当前安装路径可用空间足够，用户需进行确认。不确认，则重新选择安装路径，确认，则进入下一步骤。

步骤 5：安装小结

安装程序将打印用户之前输入的部分安装信息。如下图所示：

```
安装前小结
安装位置: /home/dmdba/dmdbms
所需空间: 733M
可用空间: 7963M
版本信息: 企业版
有效日期: 无限制
安装类型: 典型安装
是否确认安装 (Y/y,N/n) [Y/y]:y
```

图 1.44 安装小结

用户对安装信息进行确认。不确认，则退出安装程序，确认，进行 DM 的安装。

步骤 6：安装

```
2016-08-09 04:20:27
[INFO] 安装达梦数据库...
2016-08-09 04:20:27
[INFO] 安装 default 模块...
2016-08-09 04:20:37
[INFO] 安装 server 模块...
2016-08-09 04:20:37
[INFO] 安装 client 模块...
2016-08-09 04:20:47
[INFO] 安装 drivers 模块...
2016-08-09 04:20:47
[INFO] 安装 manual 模块...
2016-08-09 04:20:47
[INFO] 安装 service 模块...
2016-08-09 04:20:52
[INFO] 移动ant日志文件。
2016-08-09 04:20:52
[INFO] 安装达梦数据库完成。

请以root系统用户执行命令:
mv /home/dmdba/dmdbms/bin/dm_svc.conf /etc/dm_svc.conf

安装结束
```

图 1.45 安装过程

注意：安装完成后，终端提示“请以 root 系统用户执行命令”。由于使用非 root 系统用户进行安装，所以部分安装步骤没有相应的系统权限，需要用户手动执行相关命令。用户可根据提示完成相关操作。

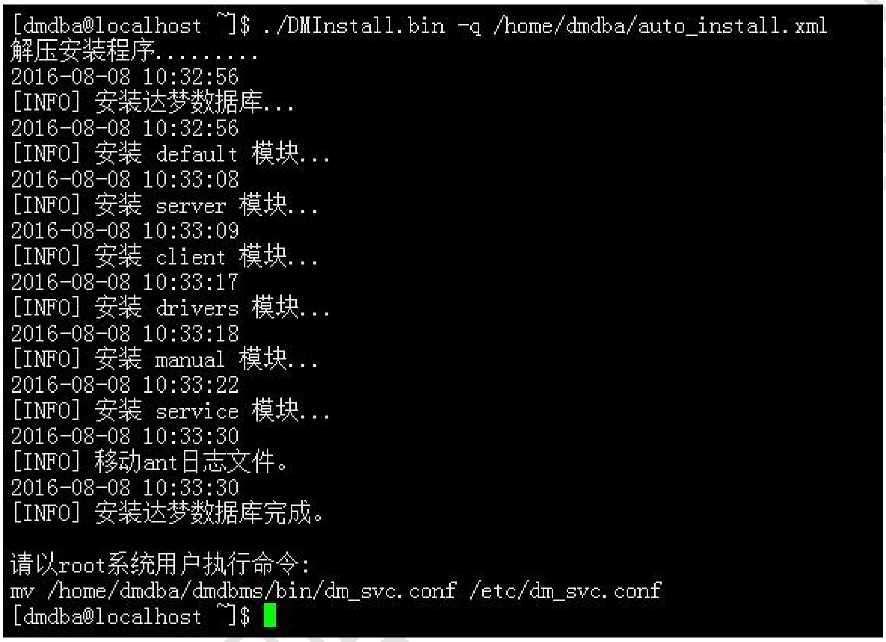
步骤 7：初始化数据库与注册服务

安装结束后，还需要初始化数据库并注册相关服务才能正式运行达梦数据库，具体可参考《DM8_dminit 使用手册》和《DM8_Linux 服务脚本使用手册》。

3） 静默安装

在某些特殊应用场景，用户可能需要非交互式的、通过配置文件进行 DM 的安装，这种情况我们可以采用以下静默安装的方式。在终端进入到安装程序所在文件夹，执行以下命令。如下图所示。

```
./DMInstall.bin -q 配置文件全路径
```


```
[dmdba@localhost ~]$ ./DMInstall.bin -q /home/dmdba/auto_install.xml
解压安装程序.........
2016-08-08 10:32:56
[INFO] 安装达梦数据库...
2016-08-08 10:32:56
[INFO] 安装 default 模块...
2016-08-08 10:33:08
[INFO] 安装 server 模块...
2016-08-08 10:33:09
[INFO] 安装 client 模块...
2016-08-08 10:33:17
[INFO] 安装 drivers 模块...
2016-08-08 10:33:18
[INFO] 安装 manual 模块...
2016-08-08 10:33:22
[INFO] 安装 service 模块...
2016-08-08 10:33:30
[INFO] 移动ant日志文件。
2016-08-08 10:33:30
[INFO] 安装达梦数据库完成。

请以root系统用户执行命令:
mv /home/dmdba/dmdbms/bin/dm_svc.conf /etc/dm_svc.conf
[dmdba@localhost ~]$
```

图 1.46 静默安装

备注：静默安装完成后，终端提示“请以 root 系统用户执行命令”。 由于使用非 root 系统用户进行安装，所以部分安装步骤没有相应的系统权限，需要用户手动执行相关命令。用户可根据提示完成相关操作。

1.1.3 卸载 DM

DM 提供的卸载程序为全部卸载。Linux 提供两种卸载方式，一种是图形化卸载方式，另一种是命令行卸载方式。

1） 图形化卸载

用户在 DM 安装目录下，找到卸载程序 uninstall.sh 来执行卸载。用户执行以下命令启动图形化卸载程序。

```
#进入 DM 安装目录
cd /DM_INSTALL_PATH
#执行卸载脚本
./uninstall.sh
```

卸载步骤：

步骤 1：运行卸载程序

程序将会弹出提示框确认是否卸载程序。点击“确定”进入卸载小结页面，点击“取消”退出卸载程序。如下图所示：

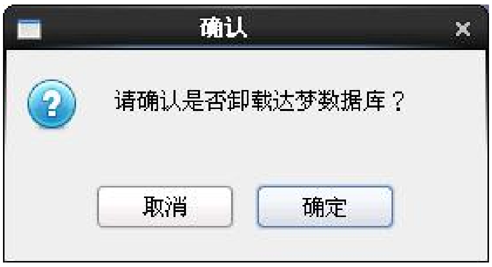


图 1.47 确认卸载

步骤 2：卸载小结

显示 DM 的卸载目录信息。点击“卸载”，开始卸载 **DM**。如下图所示。



图 1.48 准备卸载

步骤 3：卸载

显示卸载进度。如下图所示：



图 1.49 卸载

在 Linux(Unix)系统下，使用非 root 用户卸载完成时，将会弹出对话框，提示使用 root 执行相关命令，用户可根据对话框的说明完成相关操作，之后可关闭此对话框。如下图所示：

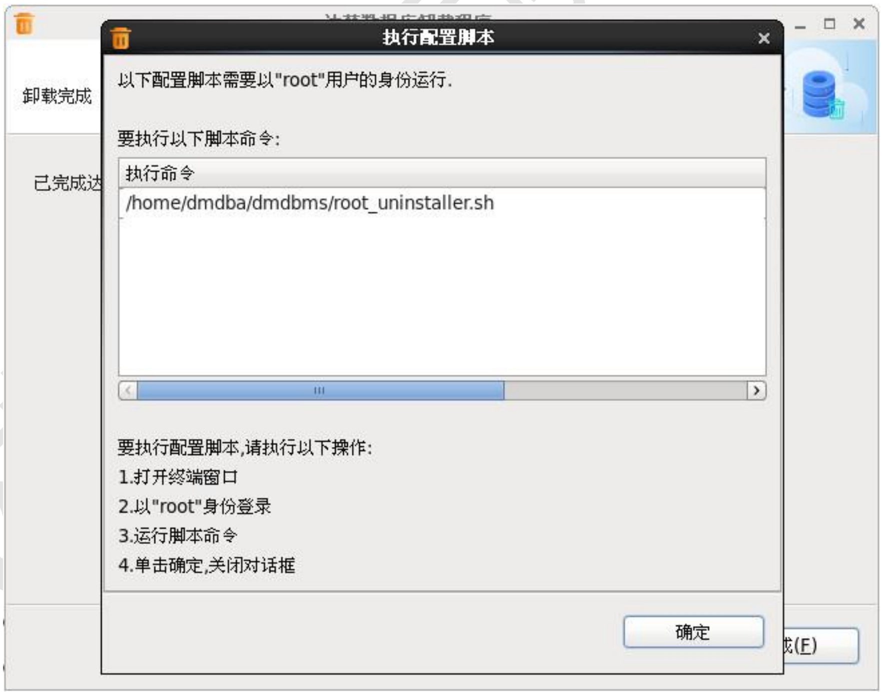


图 1.50 执行配置脚本

点击“完成”按钮结束卸载。卸载程序不会删除安装目录下有用户数据的库文件以及安装 DM 后使用过程中产生的一些文件。用户可以根据需要手工删除这些内容。如下图所示：



图 1.51 卸载完成

2） 命令行卸载

用户在 DM 安装目录下，找到卸载程序 uninstall.sh 来执行卸载。用户执行以下命令启动命令行卸载程序。

```
#进入 DM 安装目录
cd /DM_INSTALL_PATH
#执行卸载脚本命令行卸载需要添加参数-i
./uninstall.sh -i
```

卸载步骤：

步骤 1：运行卸载程序

终端窗口将提示确认是否卸载程序，输入“y/Y”开始卸载 DM，输入“n/N”退出卸载程序。如下图所示：



图 1.52 运行卸载程序

步骤 2：卸载

显示卸载进度。如下图所示：

```
[dmdba@localhost dmdbms]$ ./uninstall.sh -i
请确认是否卸载达梦数据库 [y/Y 是 n/N 否]: y

正在删除快捷方式
删除快捷方式完成
正在删除所有数据库库服务
删除数据库服务DmInstanceMonitor
删除数据库服务DmJobMonitor
删除数据库服务DmAPService
删除数据库服务DmAuditMonitor
删除所有数据库库服务完成
正在删除数据库目录
删除bin目录
删除bin目录完成
删除bin2目录
删除bin2目录完成
删除include目录
删除include目录完成
删除desktop目录
删除desktop目录完成
删除doc目录
删除doc目录完成
删除jdbc目录
删除jdbc目录完成
```

图 1.53 卸载

在 Linux(Unix)系统下，使用非 root 用户卸载完成时，终端提示“请以 root 系统用户执行命令”。用户需要手动执行相关命令。如下图所示：

```
使用root用户执行命令:
/home/dmdba/dmdbms/root_all_service_uninstaller.sh
rm -f /etc/dm_svc.conf
```

第 2 章　创建和配置 DM 数据库

DM 数据库可以在安装 DM 软件时创建，也可以在安装 DM 软件之后，通过数据库配置工具或 dminit 来手工创建数据库，创建数据库时要使用初始化参数。

建库完成后，在 DM 服务器运行期间，可以通过查询 V$DM_INI 动态视图查看建库参数的具体信息。

2.1　创建 DM 数据库

用户创建数据库之前，需要规划数据库，如数据库名、实例名、端口、文件路径、簇大小、页大小、日志文件大小、SYSDBA 和 SYSAUDITOR 等系统用户的密码等，然后可以使用图形化界面或者 dminit 创建数据库。用户可以在安装 DM 数据库软件时创建数据库，也可

达梦数据库信创大赛资料

以在安装之后创建数据库。

在创建数据库之前需要做如下准备工作：

1. 规划数据库表和索引，并估算它们所需的空间大小；
2. 确定字符集。所有字符集数据，包括数据字典中的数据，都被存储在数据库字符集中，用户在创建数据库时可以指定数据库字符集，如不指定则使用默认字符集GB18030；
3. 规划数据库文件的存储路径，可以指定数据库存储路径、控制文件存放路径、日志文件存放路径等，应注意在指定的路径或文件名中尽量不要包含中文字符，否则可能由于数据库与操作系统编码方式不一致导致不可预期的问题；
4. 配置数据库时区，如中国是+8:00 时区；
5. 设置数据库簇大小、页大小、日志文件大小，在数据库创建时由 EXTENT_SIZE、PAGE_SIZE、LOG_SIZE 初始化参数来指定，并且在数据库创建完成之后不能修改此参数。

创建数据库之前，必须满足以下必要条件：

1. 安装必需的 DM 软件，包括为操作系统设置各种环境变量，并为软件和数据库文件建立目录结构；
2. 必须有足够的内存来启动 DM 数据库实例；
3. 在执行 DM 的计算机上要有足够的磁盘存储空间来容纳规划的数据库。

2.2 使用数据库配置工具创建数据库

数据库配置工具提供了一个图形化界面来引导用户创建和配置数据库。使用数据库配置工具来创建数据库是用户的首选，因为数据库配置工具更趋于自动化，当使用数据库配置工具安装完成时，数据库也做好了使用准备工作。数据库配置工具可以通过 DM 数据库安装之后作为一个独立的工具来启动。数据库配置工具可以执行以下任务：

1. 创建数据库；
2. 改变数据库的配置；
3. 删除数据库；
4. 管理模板；
5. 自动存储管理；
6. 注册数据库服务。

2.2.1 启动数据库配置助手

数据库配置工具可以通过 DM 支持的模板或用户自定义的模板来创建数据库。

通过下列步骤来启动数据库配置助手：

1. 使用通过验证的具有管理权限组的成员登录到计算机上，安装 DM 数据库软件，并创建和运行数据库；
2. 在 linux 操作系统中启动数据库配置助手。用 dmdba 账户进入到$DM_HOME/tool 路径，运行./dbca.sh 。

达梦数据库信创大赛资料



图 2.1 启动数据库配置工具

2.2.2 使用数据库配置工具创建数据库

在数据库配置工具操作窗口中，选择“创建数据库”选项启动能够创建和配置一个数据库的向导，如图 2.2 所示。这个向导引导用户完成以下操作：

1. 数据库模板；
2. 数据库目录；
3. 数据库标识；
4. 数据库文件；
5. 初始化参数；
6. 口令管理；
7. 创建示例库；
8. 创建摘要；
9. 创建数据库。

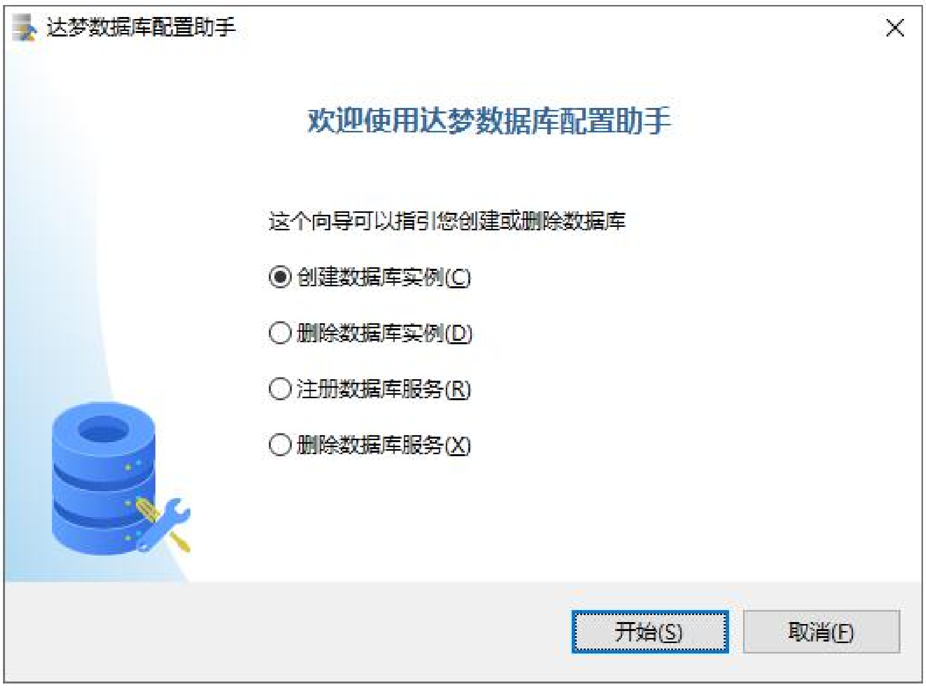


图 2.2 数据库配置工具操作界面

信创大赛资料

1） 数据库模板

在这个窗口中可以选择需要创建数据库的类型，DM 预定义了一些模板，如一般用途、联机分析处理模板或联机事务处理模板。如图 2.3 所示。

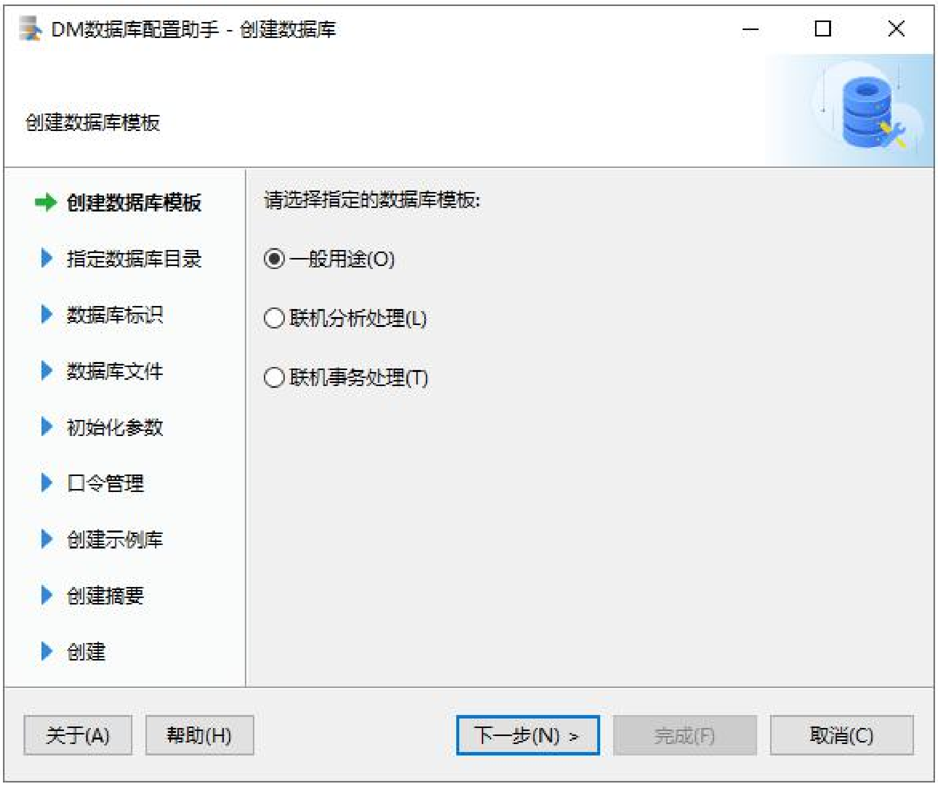


图 2.3 数据库模板选择

2） 数据库目录

指定数据库目录。如图 2.4 所示。



图 2.4 指定数据库目录

3) 数据库标识

在“数据库名（D）”文本框中，输入数据库名；在“实例名（I）”文本框中输入数据库实例名；在“端口号（P）”文本框中，输入端口号。如图 2.5 所示。

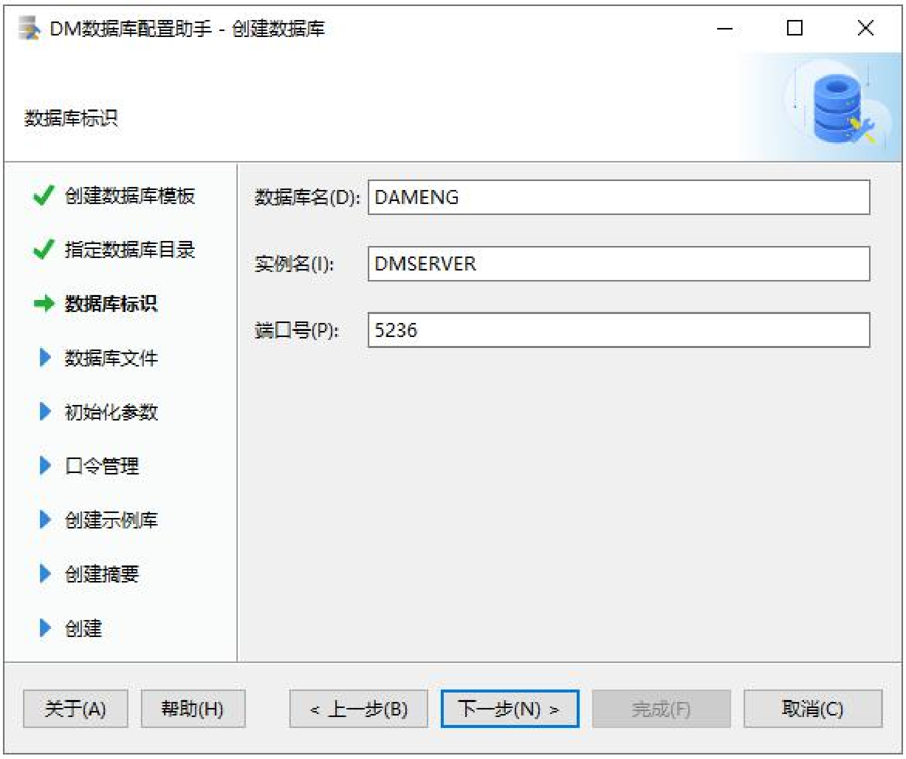


图 2.5 设置数据库标识

4） 数据库文件

如图 2.6 所示，此界面包含四个选项卡：“控制文件”、“数据文件”、“日志文件”和“初始化日志”，可以通过双击路径来更改文件路径。

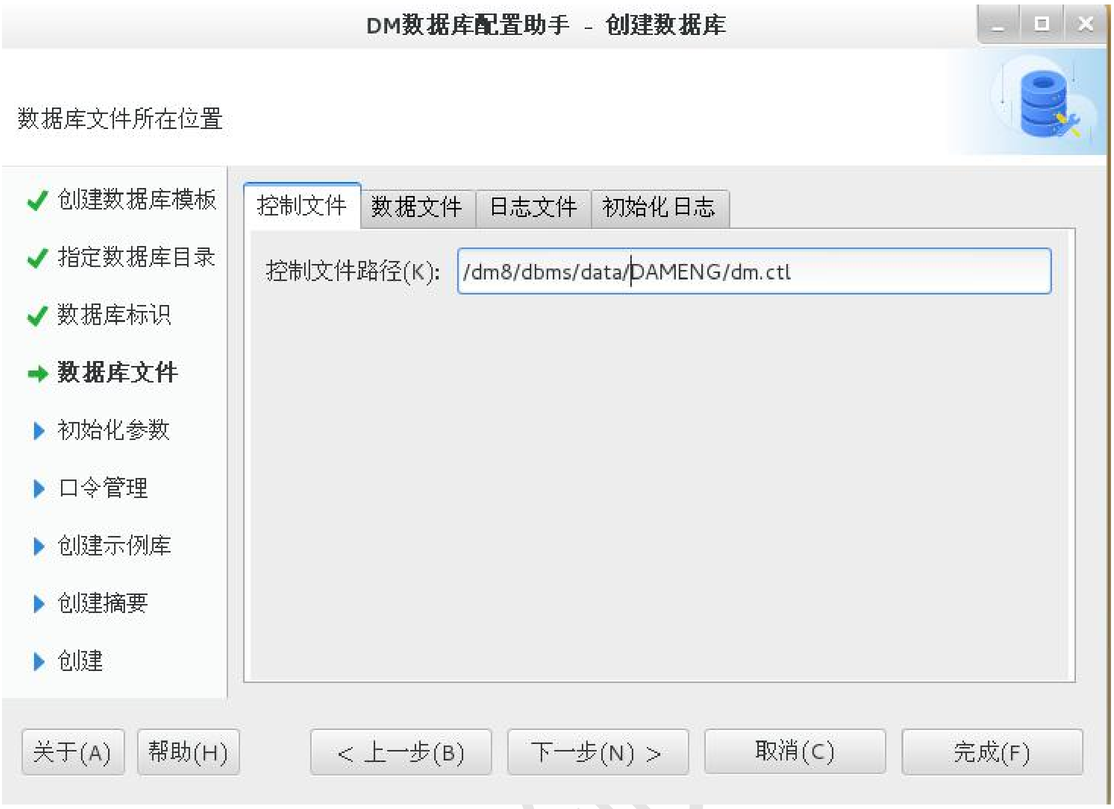


图 2.6 设置数据库文件路径

控制文件

与配置文件类似，控制文件对系统的运行及性能有很大的影响，但不同的是，配置文件中的配置项可以随意更改，而控制文件中的控制信息一般在系统第一次创建完毕后就不能随意更改，所以控制文件不是一个文本文件，而是一个二进制文件。另外控制文件一旦被破坏或丢失仍就无法手工重新创建，基于这一点，对控制文件的保护显得尤为重要。

整个系统中只有一个控制文件，其中存储的控制信息包括主要数据文件路径、日志文件路径、LSN 信息等。由于控制文件对系统至关重要，如果控制文件损坏，系统将无法启动，因此，DM 允许在创建数据库时指定多个控制文件的镜像。这些控制文件的内容是一模一样的，系统每次写控制文件时会顺序对它们进行修改。如果系统在写某一个控制文件时发生硬件故障导致该文件损坏，可以通过其他的控制文件来恢复这个损坏的控制文件，之后重新启动数据库。

数据文件

"数据文件"选项卡用来指定系统表空间路径、用户表空间路径、回滚表空间路径和临时表空间路径。同时还可以指定系统表空间镜像、用户表空间镜像、回滚表空间镜像路径。三个镜像文件是分别和系统表空间、用户表空间、回滚表空间一模一样的文件，当系统表空间、用户表空间、回滚表空间文件损坏时，就可以使用相应的镜像文件来替换。

数据文件是数据库中最重要的文件类型之一，这是数据最终要存储的地方，每个数据库至少有一个与之相关的数据文件，通常情况下，将会有多个。为了理解 DM Server 如何组织这些文件，以及数据在它们内部是如何组织的，必须理解数据页和簇的概念，它们都是 DM Server 用于保存数据库对象的分配单元。

数据页是系统进行磁盘 IO 和缓冲区调度的单元，其大小在数据库创建时就固定下来了，而且一旦固定就不可更改，它们的容量也都是相同的。所有数据页的格式大致相同。

簇是数据文件中一个连续的分配空间，簇由多个数量固定的数据页组成。数据文件对空

间的标识都以簇为单位，每个数据文件都维护着两条链，一条为半空簇的链，另一条为自由簇的链，其中半空链用于标识文件中所有被用过一部分的簇，自由链则标识文件中所有未被用过的簇。通常情况下，系统在分配空间时，以簇为单位分配会更有效。

日志文件

重做日志文件对于 DM Server 是至关重要的。它们用于存储数据库的事务日志，以便系统在出现系统故障和介质故障时能够进行故障恢复。在 DM Server 中，任何修改数据库的操作都会产生重做日志，这样，当系统出现故障时，通过分析日志可以知道在故障发生前系统做了哪些动作，并可以重做这些动作使系统恢复到故障之前的状态。

初始化日志

初始化日志用来指定初始化过程中生成的日志文件所在路径。

5) 初始化参数

数据文件使用的簇大小，即每次分配新的段空间时连续的页数，只能是 16 页或 32 页或 64 页，缺省使用 16 页。

数据文件使用的页大小，可以为 4K、8K、16K 或 32K，选择的页大小越大，则 DM 支持的元组长度也越大，但同时空间利用率可能下降，缺省使用 8K。

日志文件使用的大小，默认是 64，范围为 64 和 2048 之间的整数，单位为 M。如图 6.7 所示。

时区设置，默认是+08:00，范围为-12:59 和+14:00 之间，如图 2.7 所示。

页面检查，默认是不启用，选项包括不启用、简单检查、严格检查，如图 2.7 所示。

字符集，默认是 GB18030，选项包括 GB18030、Unicode、EUC-KR，如图 2.7 所示。

达梦数据库信创大赛资料

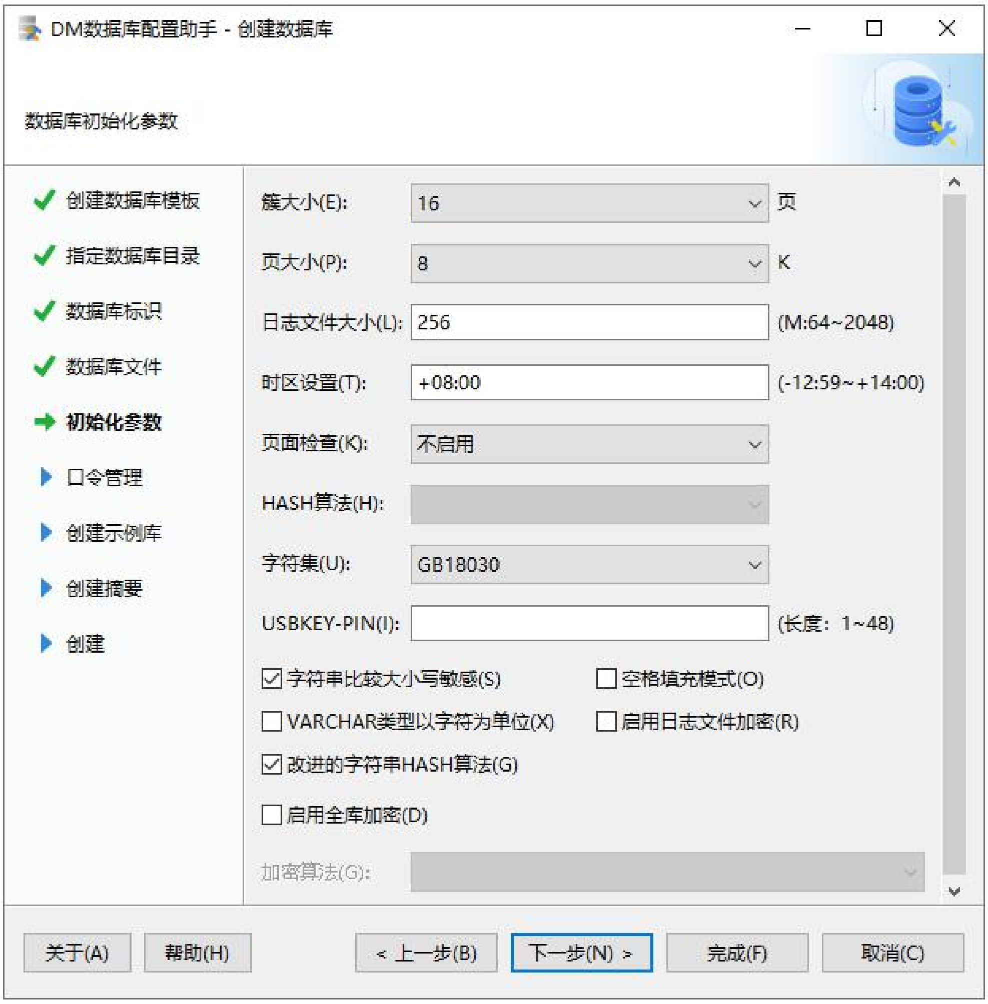


图 2.7 数据库初始化参数

6) 口令管理

为了数据库管理安全，提供了为数据库的 SYSDBA 和 SYSAUDITOR 系统用户指定新口令功能，如果安装版本为安全版，将会增加 SYSSSO 和 SYSDBO 用户的密码修改。用户可以选择为每个系统用户设置不同口令（留空表示使用默认口令），也可以为所有系统用户设置同一口令。口令必须是合法的字符串，不能少于 9 个或多于 48 个字符。如图 2.8 所示。

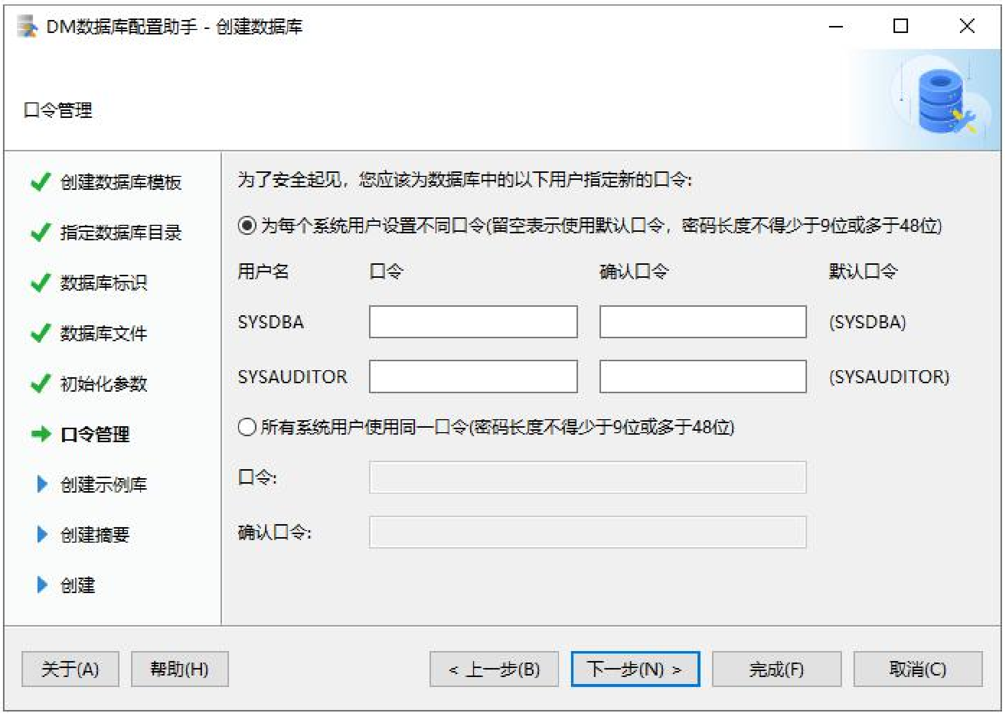


图 2.8 口令管理

7） 创建示例库

示例库 BOOKSHOP 模拟武汉代理图书的某销售公司，该公司欲建立在线购物平台来拓展其代理产品的销售渠道。该示例在 DM 各演示程序中使用。安装该示例后，将在数据库中创建 BOOKSHOP 表空间，同时创建 RESOURCES、PERSON、PRODUCTION、PURCHASING、SALES、OTHER 这 6 个模式和相关的表。

示例库 DMHR 模拟武汉达梦数据库有限公司人力资源管理系统。安装完该示例库，将创建一个模式 DMHR 和一个表空间 DMHR，在 DMHR 模式下创建 REGION、CITY、LOCATION、DEPARTMENT、JOB、EMPLOYEE、JOB_HISTORY 等 7 张表，并插入数据。

如图 2.9 所示。

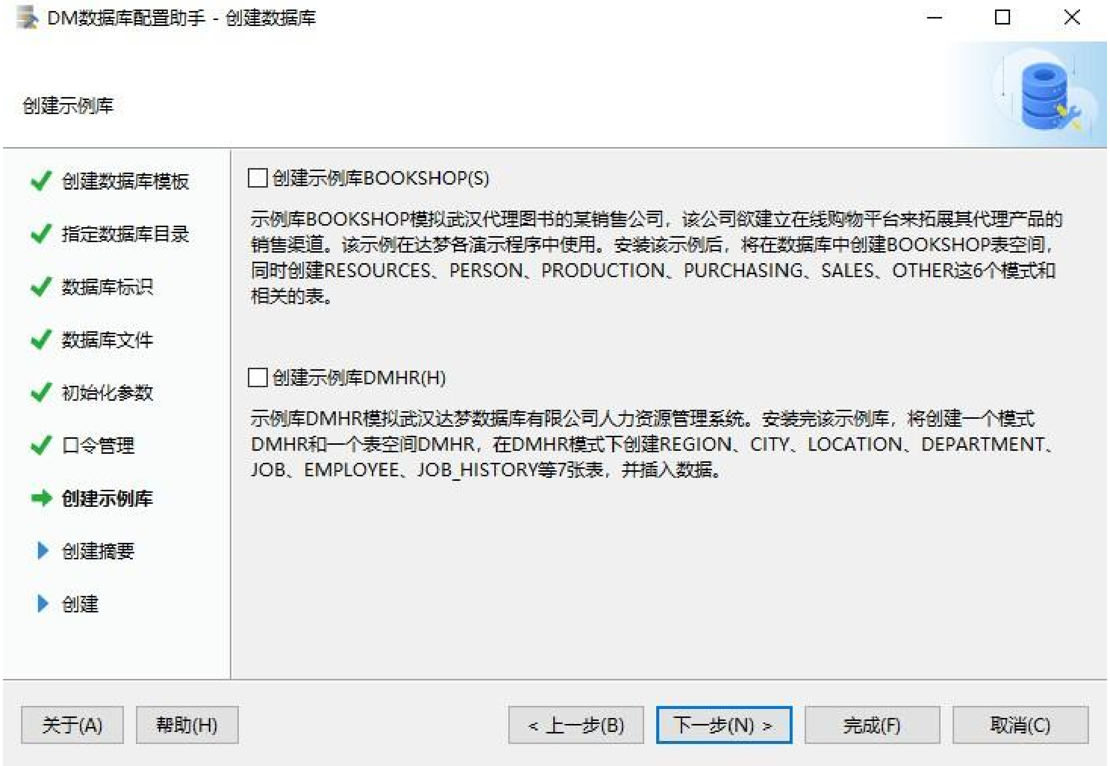


图 2.9 创建示例库

8） 创建摘要

列举创建数据库纲要，会列举创建时指定的数据库名、示例名、数据库目录、端口、控制文件路径、数据文件路径、日志文件路径、簇大小、页大小、日志文件大小、标识符大小写是否敏感、是否使用 unicode 等信息，方便用户确认创建信息是否符合自己的需求，及时返回修改。如图 2.10 所示。



图 2.10 创建摘要

9） 创建数据库

核对完创建信息后，开始创建数据库、创建并启动实例、创建示例库。如图 2.11 所示。

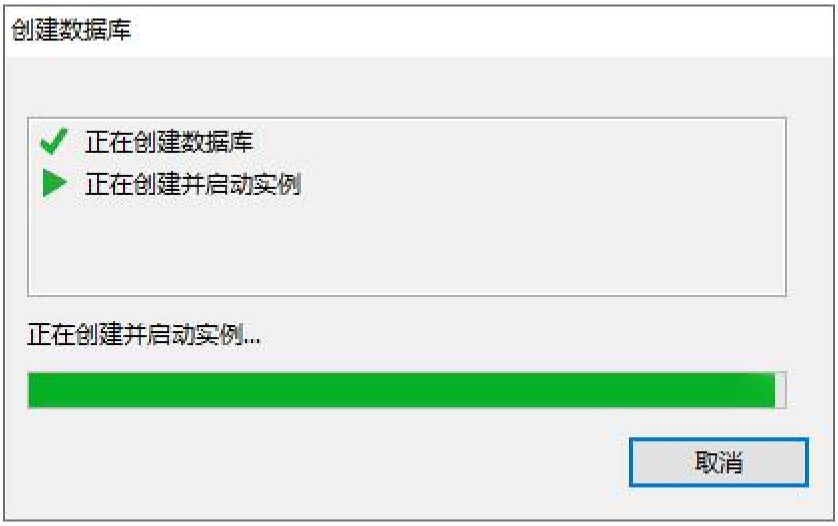


图 2.11 创建数据库

安装完成之后将显示对话框，提示安装完成或错误反馈信息，如图 2.12 所示。



图 2.12 创建数据库完成

数据库配置工具运行在 Linux、Solaris、AIX 和 HP-UNIX 系统中，使用非 root 系统用户在创建数据库完成时，将弹出提示框，提示应以 root 系统用户执行以下命令，用来创建数据库的随机启动服务，如图 2.13 所示。



图 2.13 执行命令提示

2.3 使用 dminit 创建数据库

在安装 DM 的过程中，用户可以选择是否创建初始数据库，如果当时没有创建，也可以在完成安装后，利用初始化库工具 dminit 来创建。系统管理员可以利用该工具提供的各种参数，设置数据库存放路径、段页大小、是否对大小写敏感以及是否使用 unicode，创建出

满足用户需要的初始数据库。该工具位于安装目录的 bin/目录下。更多详细的功能请参考《DM8_dminit 使用手册》。

在 linux 终端中输入带参数的 dminit 命令启动 dminit 工具，命令格式为 dminit [para=value][para=value]……，参数说明见表 2.1。

表 2.1 dminit 工具参数说明

参数	含义	取值	备注
INI_FILE	已有 INI 文件的路径,此 INI 文件用于将其所有参数值作为当前新生成 INI 文件的参数值	合法的路径。文件路径长度最大为 257(含结束符)，不包括文件名	可选
PATH	初始数据库存放的路径，默认路径为 dminit.exe 当前所在的工作目录	合法的路径。文件路径长度最大为 257(含结束符)，不包括文件名	可选
CTL_PATH	初始数据库控制文件的路径，默认值 linux 下是/**PATH**/**DM_NAME**/dm.ctl（粗体表示前面设置的参数）	合法的路径。文件路径长度最大为 257(含结束符)，不包括文件名	可选
LOG_PATH	初始数据库日志文件的路径，默认值 linux 下是 **PATH**/**DB_NAME**/DB_NAME01.log 和 **PATH**/**DB_NAME**/DB_NAME02.log(粗体表示前面设置的参数)	合法的路径。文件路径长度最大为 257(含结束符)，不包括文件名。日志文件路径个数不超过 10 个	可选
EXTENT_SIZE	数据文件使用的簇大小，即每次分配新的段空间时连续的页数	只能是 16 页或 32 页或 64 页之一，缺省使用 16 页	可选
PAGE_SIZE	数据文件使用的页大小,可以为 4K、8K、16K 或 32K 之一，选择的页大小越大，则 DM 支持的元组长度也越大，但同时空间利用率可能下降，缺省使用 8K	只能是 4K、8K、16K 或 32K 之一	可选
LOG_SIZE	日志文件使用的簇大小，以 M 为单位，默认每个日志文件大小为 256M	64 和 2048 之间的整数	可选
CASE_SENSITIVE	标识符大小写敏感，默认值为 Y。当大小写敏感时，小写的标识符应用双引号括起，否则被转换为大写；当大小写不敏感时，系统不自动转换标识符的大小写，在标识符比较时也不区分大小写	只能是 Y，y，N，n，1，0 之一	可选
CHARSET/UNICODE_FLAG	字符集选项。0 代表 GB18030,1 代表 UTF-8,2 代表韩文字符集 EUC-KR	取值 0、1 或 2 之一。默认值为 0	可选
AUTO_OVERWRITE	0 不覆盖，表示建库目录下如果没有同名文件，直接创建。如果遇到同名文件时，屏幕提示是否需要覆盖，由用户手动输入是与否（y/n，1/0）； 1 部分覆盖，表示覆盖建库目录下所有同名文件； 2 完全覆盖，表示先清理掉建库目录下	只能是 0,1,2 之一	可选

	所有文件再重新创建。 默认值为 0。		
LENGTH_IN_CHAR	VARCHAR 类型对象的长度是否以字符为单位。 1：是，设置为以字符为单位时，定义长度并非真正按照字符长度调整，而是将存储长度值按照理论字符长度进行放大。所以会出现实际可插入字符数超过定义长度的情况，这种情况也是允许的。同时，存储的字节长度 8188 上限仍然不变，也就是说，即使定义列长度为 8188 字符，其实际能插入的字符串占用总字节长度仍然不能超过 8188； 0：否，所有 VARCHAR 类型对象的长度以字节为单位	取值 0 或 1。默认值为 0	可选
USE_NEW_HASH	字符类型在计算 HASH 值时所采用的 HASH 算法类别。0：原始 HASH 算法；1：改进的 HASH 算法。默认值为 1。	取值 0 或 1	可选
SYSDBA_PWD	初始化时设置 SYSDBA 的密码，默认为 SYSDBA	合法的字符串，长度在 9 到 48 个字符之间	可选
SYSAUDITOR_PWD	初始化时设置 SYSAUDITOR 的密码，默认为 SYSAUDITOR	合法的字符串，长度在 9 到 48 个字符之间	可选
DB_NAME	初始化数据库名字，默认是 DAMENG	有效的字符串，不超过 128 个字符	可选
INSTANCE_NAME	初始化数据库实例名字，默认是 DMSERVER	有效的字符串，不超过 128 个字符	可选
PORT_NUM	初始化时设置 dm.ini 中的 PORT_NUM，默认 5236	取值范围：1024~65534	可选
BUFFER	初始化时设置系统缓存大小，单位为 M，默认为 100	取值范围：8~1024*1024	可选
TIME_ZONE	初始化时区，默认是东八区	格式为[正负号]小时[:分钟]（正负号和分钟为可选）。时区设置范围为：-12:59~+14:00	可选
PAGE_CHECK	是否启用页面内容校验，0：不启用；1：简单校验；2：严格校验(使用 CRC16 算法生成校验码)。默认 0	取值范围：0~2	可选
DSC_NODE	高性能集群的节点数目	取值范围 2~16，单站点时不写	可选
EXTERNAL_CIPHER_NAME	设置默认加密算法	有效的字符串，不超过 128 个字符	可选
EXTERNAL_HASH_NAME	设置默认 HASH 算法	有效的字符串，不超过 128 个字符	可选
EXTERNAL_CRYPTO_NAME	设置根密钥加密引擎	有效的字符串，不超过 128 个字符	可选
ENCRYPT_NAME	全库加密密钥使用的算法名。算法可以是 DM 内部支持的加密算法，或者是第	合法的字符串，最长为 128 个字节	可选

	三方的加密算法。默认使用"AES256_ECB"算法加密		
RLOG_ENC_FLAG	设置联机日志文件和归档日志文件是否加密	取值 Y/N，y/n，1/0，默认 N	可选
USBKEY_PIN	USBKEY PIN,用于加密服务器根密钥	合法的字符串,最长为 48 个字节	可选
BLANK_PAD_MODE	设置字符串比较时，结尾空格填充模式是否兼容 ORACLE	取值 0 或 1。0 不兼容，1 兼容。默认为 0	可选
SYSTEM_MIRROR_PATH	指定 system.dbf 文件的镜像路径	绝对路径，默认为空	可选
MAIN_MIRROR_PATH	指定 main.dbf 文件的镜像路径	绝对路径，默认为空	可选
ROLL_MIRROR_PATH	指定 roll.dbf 文件的镜像路径	绝对路径，默认为空	可选
MAL_FLAG	初始化时设置 dm.ini 中的 MAL_INI，默认 0	取值 0 或 1	可选
ARCH_FLAG	初始化时设置 dm.ini 中的 ARCH_INI，默认 0	取值 0 或 1	可选
MPP_FLAG	Mpp 系统内的库初始化时设置 dm.ini 中的 mpp_ini，默认 0。	取值 0 或 1	可选
CONTROL	指定初始化配置文件路径。初始化配置文件是一个保存了各数据文件路径设置等信息的文本。使用 control 初始化时，若文件已存在，系统会屏幕打印提示，然后直接覆盖	主要用于将数据文件放在裸设备或 DSC 环境下	可选
DCP_MODE	是否是 DCP 代理模式	取值：1 是；0 否。默认值为 0	可选
DCP_PORT_NUM	DCP 代理模式下管理端口	取值范围：1024~65534。当 DCP_MODE=1 时，该参数才有效	可选
SYSSSO_PWD	初始化时设置 SYSSSO 的密码，默认为 SYSSSO,仅在安全版本下可见和可设置	合法的字符串，长度在 6 到 48 个字符之间	可选
SYSDBO_PWD	初始化时设置 SYSDBO 的密码，默认为 SYSDBO,仅在安全版本下可见和可设置	合法的字符串，长度在 6 到 48 个字符之间	可选
PRIV_FLAG	是否是四权分立。默认值为 0（不使用），四权分立的具体权限见《DM8 安全管理》。默认情况下，使用三权分立。仅在安全版本下可见和可设置	只能是 0 或 1	可选
ELOG_PATH	指定初始化过程中生成的日志文件所在路径	合法的路径。文件路径长度最大为 257(含结束符)，不包括文件名	可选
PAGE_ENC_SLICE_SIZE	数据页加密分片大小	可配置大小为 512 或 4096。默认为 0，表示兼容老数据库的页加密	可选
HUGE_WITH_DELTA	是否仅支持创建事务型 HUGE 表	取值：1 是；0 否。默认值为 1	可选
RLOG_GEN_FOR_HUGE	是否生成 HUGE 表 REDO 日志	取值：1 是；0 否。默认	可选

		值为 0	
PSEG_MGR_FLAG	是否仅使用管理段记录事务信息	取值：1 是；0 否。默认值为 0	可选
HELP	显示帮助信息		可选

说明：dminit 一般是要有参数的，如果没有带参数，系统就会引导用户设置。另外，参数、等号和值之间不能有空格。Help 参数的后面不用添加'='号。

命令举例：

```
dminit PATH=/dm8/dmdbms PAGE_SIZE=16
```

如果创建成功，则屏幕显示如图 2.14 所示。此时在/dm8 目录下出现 dmdbms 文件夹，内容包含初始数据库 DAMENG 的相关文件和初始化文件 dm.ini。将 dm.ini 文件拷贝到 DM 安装目录的 bin 下，DM 服务器就可以启动该初始数据库了。

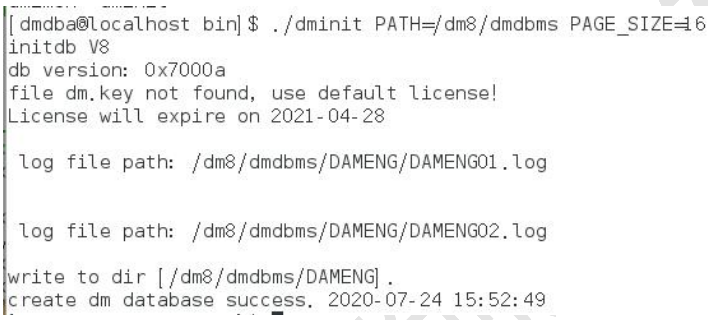


图 2.14 创建成功后的屏幕显示

2.4 注册数据库服务

注册数据库服务，该功能用于将使用命令行工具生成的数据库，重新注册成系统服务，方便用户管理与控制。

用户注册数据库服务可以通过图形化界面和 shell 脚本两种方式,本节讲述的是通过数据库配置工具注册数据库服务，通过 shell 脚本注册数据库服务详细操作请参见《DM8_Linux 服务脚本手册》。如图 2.15 所示。

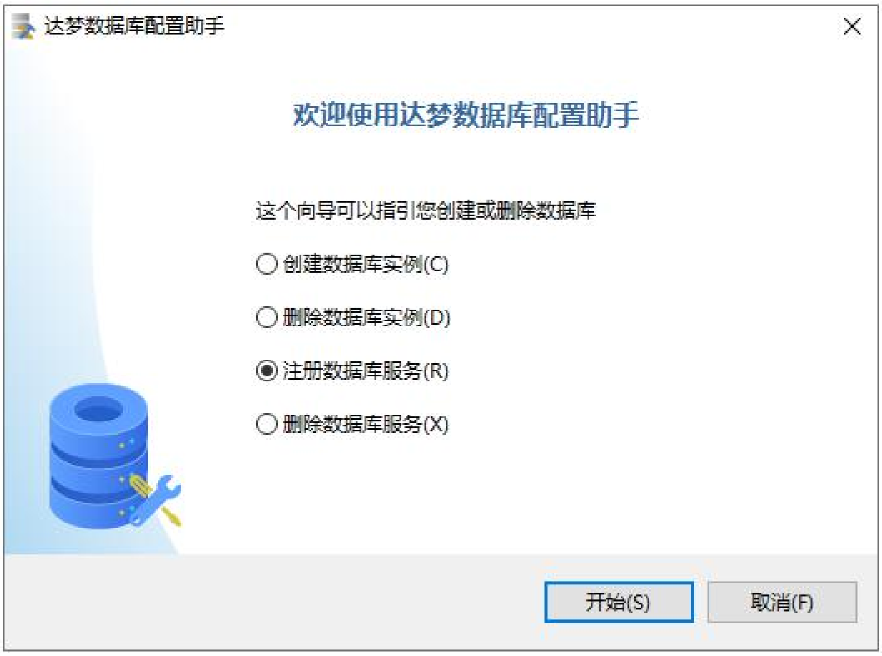


图 2.15 选择操作方式

用户选择 dm.ini 文件来注册相应的数据库，并可修改相应的端口号和实例名，也可以选择是否以配置状态启动数据库，如图 2.16 所示。



图 2.16 注册数据库服务

正在注册数据库服务，如图 2.17 所示。

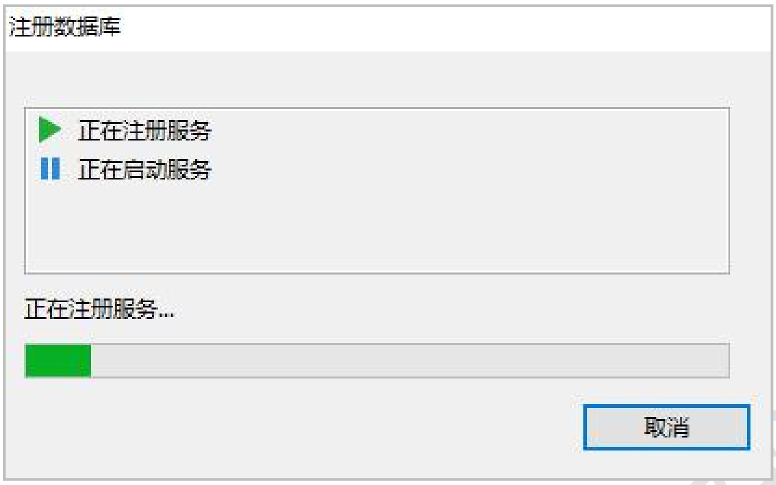


图 2.17 注册数据库服务

注册完成之后将显示对话框，提示完成或错误反馈信息，如图 2.18 所示。



图 2.18 注册数据库完成

2.5 查看数据库信息

登录 DM 管理工具，通过表空间属性，可以查看数据文件路径、总空间大小、空闲空间大小、使用率。如图 2.19 所示。

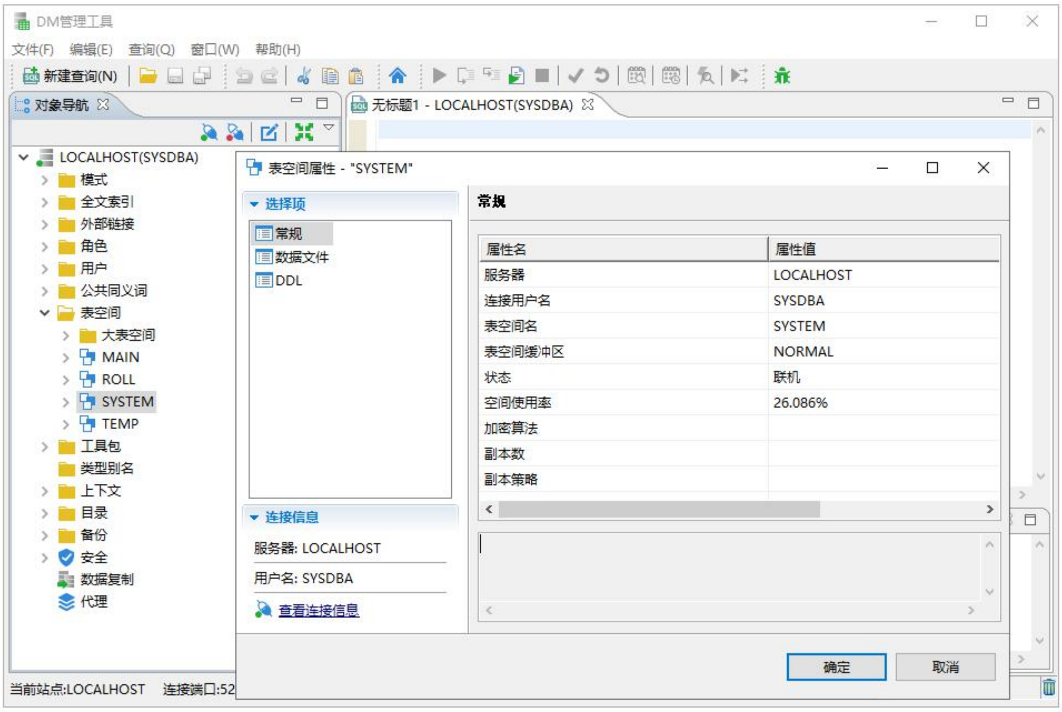


图 2.19 查看数据文件

通过 DM 控制台工具，查看实例配置属性，如图 2.20 所示，可以查看到如下信息：文件位置、实例名、内存池和缓冲区、线程、查询、检查点、输入输出、数据库、日志、事务、安全、兼容性、请求跟踪、进程守护、ORACLE 数据类型兼容和配置文件。

达梦数据库信创大

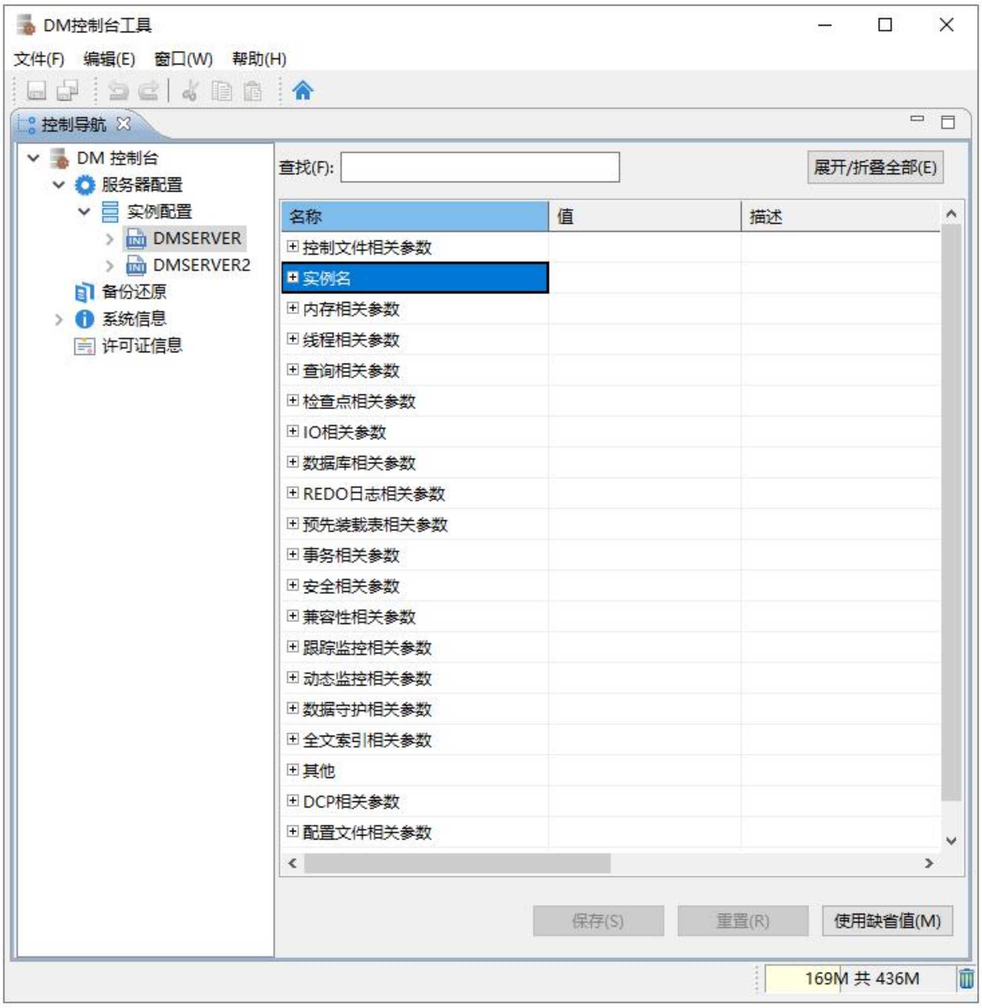


图 2.20 查看数据文件

2.6 删除数据库

删除数据库，包括删除数据库的数据文件、日志文件、控制文件和初始化参数文件。为了保证删除数据库成功，必须保证 dmserver 已关闭。

可以使用数据库配置工具来删除数据库。如图 2.21 所示。

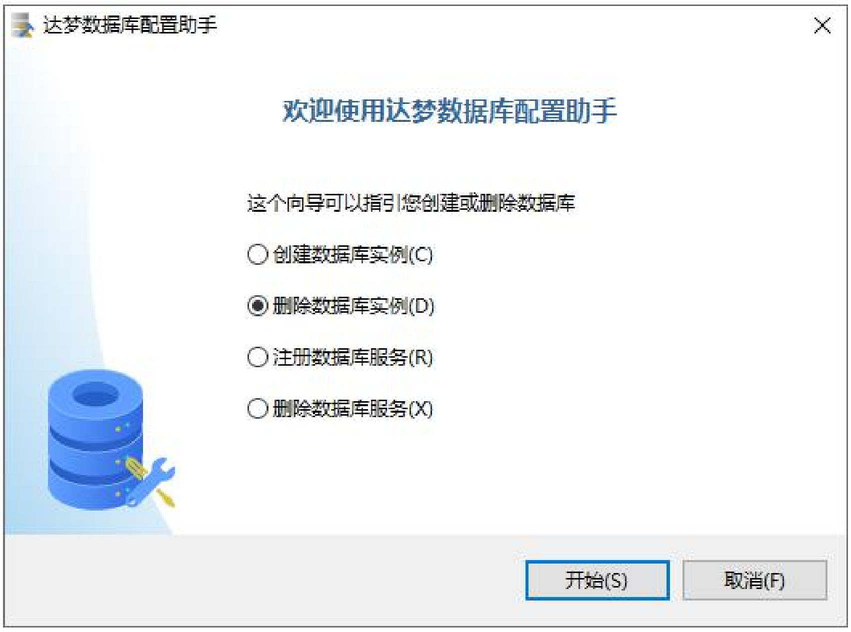


图 2.21 选择操作方式

根据数据库名称，选择要删除的数据库，如图 2.22 所示，也可以通过指定数据库配置文件删除数据库。

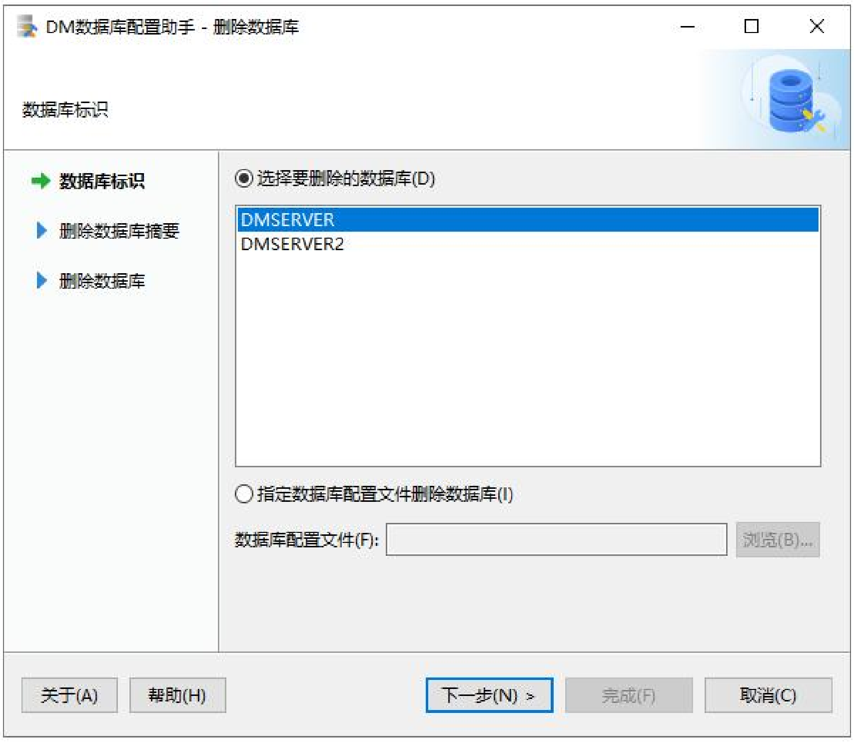


图 2.22 要删除的数据库

确认将删除的数据库名、实例名、数据库目录，如图 2.23 所示。



图 2.23 删除数据库概要

首先停止实例，然后删除实例，最后删除数据库，如图 2.24 所示。

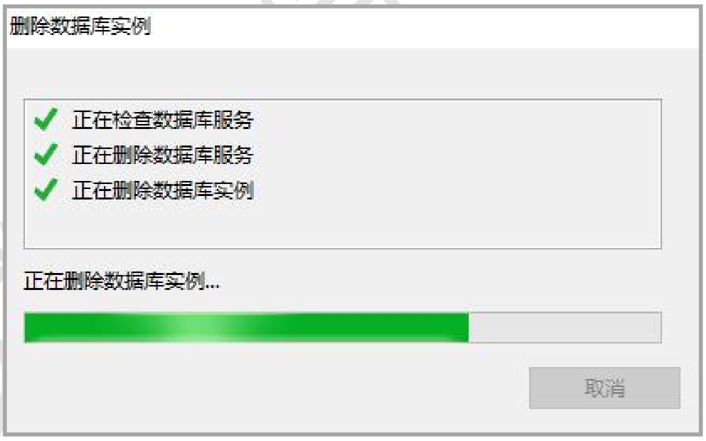


图 2.24 删除数据库

删除完成之后将显示对话框，提示完成或错误反馈信息，如图 2.25 所示。



图 2.25 删除数据库完成

如果数据库配置工具运行在 Linux、Solaris、AIX 和 HP-UNIX 系统中，使用非 root 系统用户在删除数据库完成时，将弹出提示框，提示应以 root 系统用户执行以下命令，用来删除数据库的随机启动服务，如图 2.26 所示。



图 2.26 执行命令提示

2.7 删除数据库服务

删除数据库服务，只删除用于启动和停止数据库的服务文件，不会删除数据库的数据文件、日志文件、控制文件和初始化参数文件。

用户删除数据库服务可以通过图形化界面和 shell 脚本两种方式，本节讲述的是通过数据库配置工具删除数据库服务如图 2.27 所示。

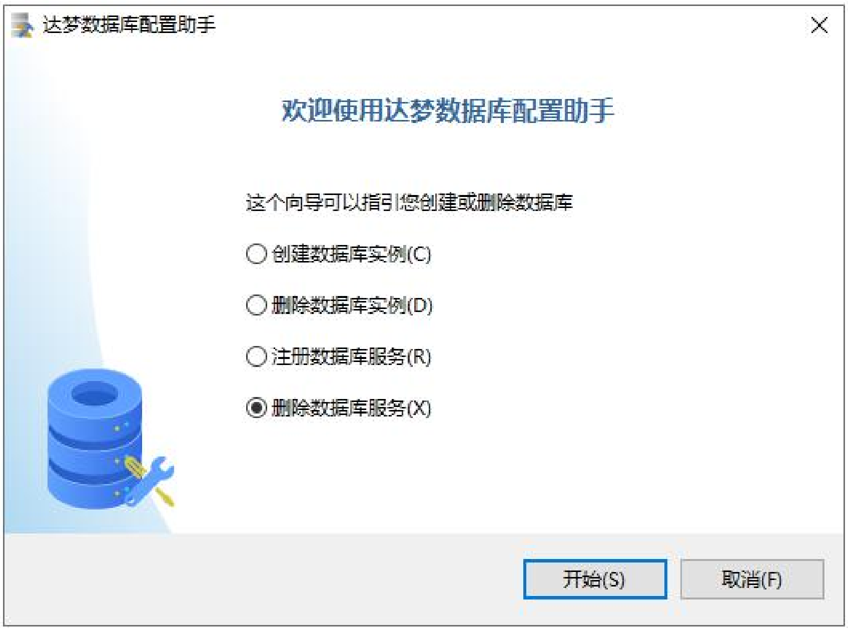


图 2.27 选择操作方式

根据数据库服务名称，选择要删除的数据库服务，如图 2.28 所示，也可以通过指定数据库配置文件删除数据库服务。

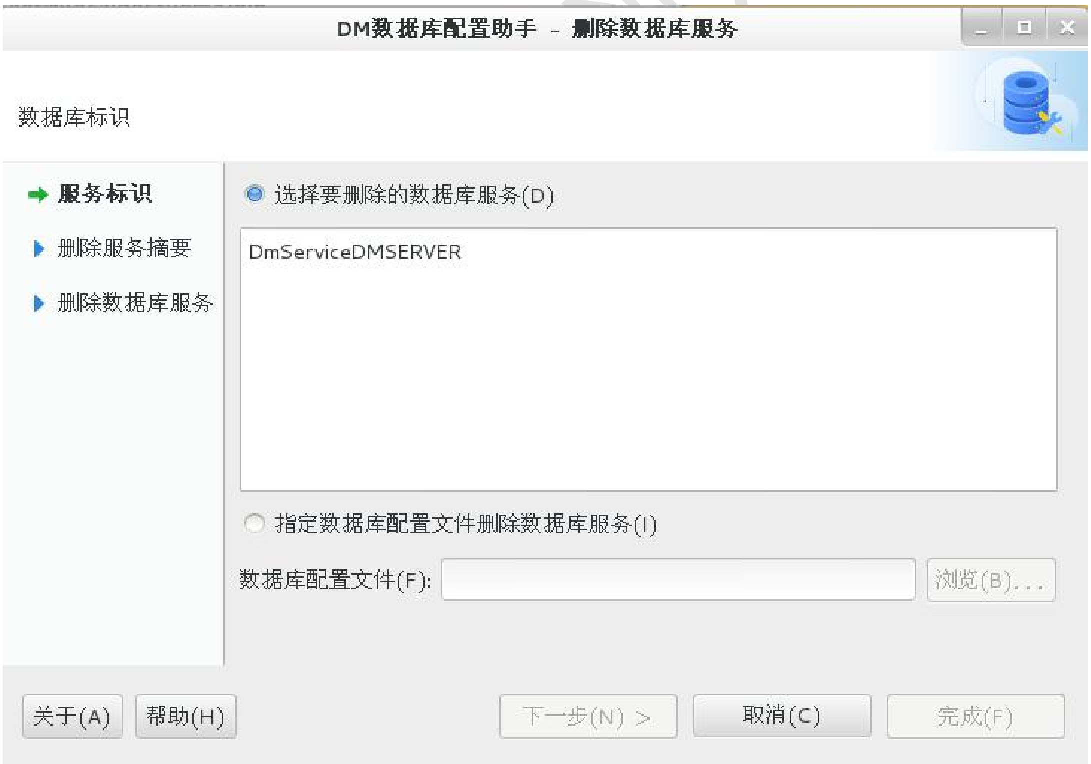


图 2.28 要删除的数据库服务

确认将删除的数据库名、实例名、数据库服务名、数据库目录，如图 2.29 所示。



图 2.29 删除数据库概要

首先检查数据库服务，然后删除数据库服务，如图 2.30 所示。

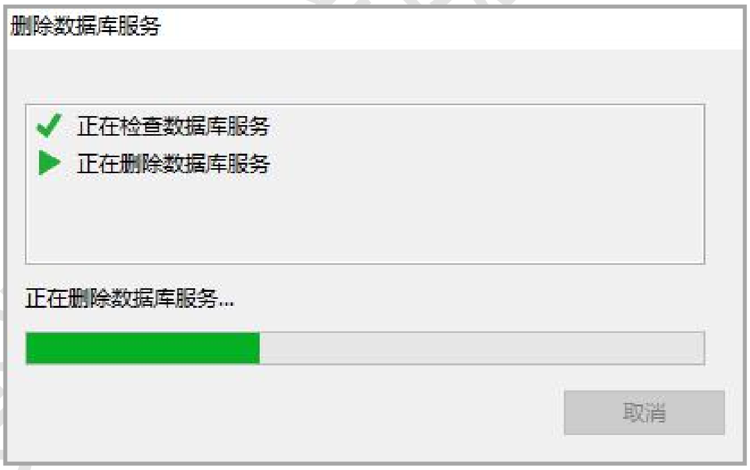


图 2.30 删除数据库服务

删除完成之后将显示对话框，提示完成或错误反馈信息，如图 2.31 所示。



图 2.31 删除数据库服务完成

如果数据库配置工具运行在 Linux、Solaris、AIX 和 HP-UNIX 系统中，使用非 root 系统用户在删除数据库服务完成时，将弹出提示框，提示应以 root 系统用户执行以下命令，用来删除数据库的随机启动服务，如图 2.32 所示。



图 2.32 执行命令提示

第 3 章　启动和关闭数据库

DM 支持多种操作系统。常用的操作系统基本上可以划分为 Windows 操作系统和 Linux 操作系统。下面分别介绍这两种操作系统下启动和关闭 DM 数据库的操作。

3.1　启动数据库

3.1.1 Linux 系统

1. 菜单方式

安装 DM 数据库后(默认情况下安装成功后 DM 服务会自动启动)，在 Linux 的开始菜单选项中选择启动服务器菜单项可以启动 DM 数据库。启动方式类似 Windows。

2. Linux 服务方式

安装 DM 数据库后，在/etc/rc.d/init.d 中有名称为 DmService 开头的文件，文件全名为 DmService+实例名（例：如果实例名为 DMSERVER，则相对应的服务文件为 DmServiceDMSERVER）。以实例名为 DMSERVER 为例，在终端输入./DmServiceDMSERVER start 或者 service DmServiceDMSERVER start 即可启动 DM 数据库。

3. 命令行方式

在终端进入 DM 安装目录下的 bin 目录，执行./dmserver 启动 DM 数据库，参数选项同 Windows。

3.1.2 检查 LICENSE

无论是在何种操作系统下运行，DM 数据库在启动时都会进行 LICENSE 检查。若 LICENSE 过期或 KEY 文件与实际运行环境不配套，DM 服务器会强制退出。

可通过查看 V$LICENSE 了解所安装的 DM 数据库的 LICENSE 信息。

3.2　数据库状态和模式

DM 数据库包含以下几种状态：

1. 配置状态（MOUNT）：不允许访问数据库对象，只能进行控制文件维护、归档配置、数据库模式修改等操作；
2. 打开状态（OPEN）：不能进行控制文件维护、归档配置等操作，可以访问数据库对象，对外提供正常的数据库服务；
3. 挂起状态（SUSPEND）：与 OPEN 状态的唯一区别就是，限制磁盘写入功能；一旦修改了数据页，触发 REDO 日志、数据页刷盘，当前用户将被挂起。

OPEN 状态与 MOUNT 和 SUSPEND 能相互转换，但是 MOUNT 和 SUSPEND 之间不能相互转换。

达梦数据库信创大赛资料

DM 数据库包含以下几种模式：

1. 普通模式（NORMAL）：用户可以正常访问数据库，操作没有限制；
2. 主库模式（PRIMARY）：用户可以正常访问数据库，所有对数据库对象的修改强制生成 REDO 日志，在归档有效时，发送 REDO 日志到备库；
3. 备库模式（STANDBY）：接收主库发送过来的 REDO 日志并重做。数据对用户只读。

三种模式只能在 MOUNT 状态下设置，模式之间可以相互转换。

对于新初始化的库，首次启动不允许使用 mount 方式，需要先正常启动并正常退出，然后才允许 mount 方式启动。

一般情况下，数据库为 NORMAL 模式，如果不指定 MOUNT 状态启动，则自动启动到 OPEN 状态。

在需要对数据库配置时（如配置数据守护、数据复制），服务器需要指定 MOUNT 状态启动。当数据库模式为非 NORMAL 模式（PRIMARY、STANDBY 模式），无论是否指定启动状态，服务器启动时自动启动到 MOUNT 状态。

3.3 关闭数据库

3.3.1 Linux 系统

1. 菜单方式

在 Linux 的开始菜单中选择关闭数据库的菜单项可以关闭 DM 数据库。关闭方式类似 Windows。

2. Linux 服务方式

进入/etc/rc.d/init.d，以实例名为 DMSERVER 为例，在命令行工具中输入./DmServiceDMSERVER stop 即可关闭 DM 数据库。

3. 命令行方式

在启动数据库的命令工具中输入 exit，然后回车，退出 DM 数据库。关闭方式类似 Windows。

达梦数据库信创大赛资料

第 4 章　DM 逻辑结构概述

4.1　数据库和实例

在 DM7 之前版本的 DM 数据库中，“数据库”和“实例”这两个术语经常可以互相替换，意义也很相近。在 DM7 以及之后版本的数据库中，“数据库”和“实例”这两个概念之间有着很大的差别，甚至可以说它们是两个完全不同的实体。

4.1.1 数据库

在有些情况下，数据库的概念包含的内容会很广泛。如在单独提到 DM 数据库时，可能指的是 DM 数据库产品，也有可能是正在运行的 DM 数据库实例，还可能是 DM 数据库运行中所需的一系列物理文件的集合等。但是，当同时出现 DM 数据库和实例时，DM 数据库指的是磁盘上存放在 DM 数据库中的数据的集合，一般包括：数据文件、日志文件、控制文件以及临时数据文件等。

4.1.2 实例

实例一般是由一组正在运行的 DM 后台进程/线程以及一个大型的共享内存组成。简单来

达梦数据库信创大赛资料

说，实例就是操作 DM 数据库的一种手段，是用来访问数据库的内存结构以及后台进程的集合。

DM 数据库存储在服务器的磁盘上，而 DM 实例则存储于服务器的内存中。通过运行 DM 实例，可以操作 DM 数据库中的内容。在任何时候，一个实例只能与一个数据库进行关联（装载、打开或者挂起数据库）。在大多数情况下，一个数据库也只有一个实例对其进行操作。但是在 DM 共享存储集群（DMDSC）中，多个实例可以同时装载并打开一个数据库（位于一组由多台服务器共享的物理磁盘上）。此时，我们可以同时从多台不同的计算机访问这个数据库。

4.2 DM 逻辑存储结构

DM 数据库为数据库中的所有对象分配逻辑空间，并存放在数据文件中。在 DM 数据库内部，所有的数据文件组合在一起被划分到一个或者多个表空间中，所有的数据库内部对象都存放在这些表空间中。同时，表空间被进一步划分为段、簇和页（也称块）。通过这种细分，可以使得 DM 数据库能够更加高效地控制磁盘空间的利用率。图 4.1 显示了这些数据结构之间的关系。

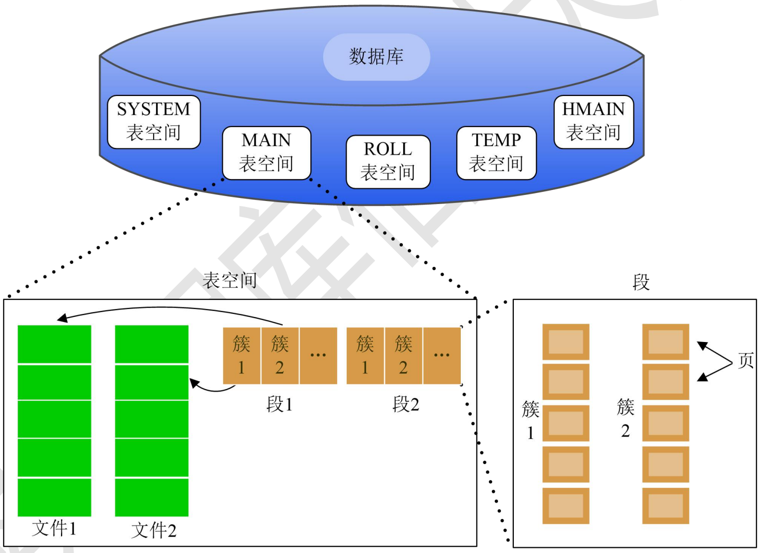


图 4.1 显示表空间、数据文件、段、簇、页的关系

可以看出，在 DM8 中存储的层次结构如下：

1. 数据库由一个或多个表空间组成；
2. 每个表空间由一个或多个数据文件组成；
3. 每个数据文件由一个或多个簇组成；
4. 段是簇的上级逻辑单元，一个段可以跨多个数据文件；
5. 簇由磁盘上连续的页组成，一个簇总是在一个数据文件中；
6. 页是数据库中最小的分配单元，也是数据库中使用的最小的 IO 单元。

4.2.1 表空间

在 DM 数据库中，表空间由一个或者多个数据文件组成。DM 数据库中的所有对象在逻辑上都存放在表空间中，而物理上都存储在所属表空间的数据文件中。

在创建 DM 数据库时，会自动创建 5 个表空间：SYSTEM 表空间、ROLL 表空间、MAIN 表空间、TEMP 表空间和 HMAIN 表空间。

1. SYSTEM 表空间存放了有关 DM 数据库的字典信息，用户不能在 SYSTEM 表空间创建表和索引。
2. ROLL 表空间完全由 DM 数据库自动维护，用户无需干预。该表空间用来存放事务运行过程中执行 DML 操作之前的值，从而为访问该表的其他用户提供表数据的读一致性视图。
3. MAIN 表空间在初始化库的时候，就会自动创建一个大小为 128M 的数据文件 MAIN.DBF。在创建用户时，如果没有指定默认表空间，则系统自动指定 MAIN 表空间为用户默认的表空间。
4. TEMP 表空间完全由 DM 数据库自动维护。当用户的 SQL 语句需要磁盘空间来完成某个操作时，DM 数据库会从 TEMP 表空间分配临时段。如创建索引、无法在内存中完成的排序操作、SQL 语句中间结果集以及用户创建的临时表等都会使用到 TEMP 表空间。
5. HMAIN 表空间属于 HTS 表空间，完全由 DM 数据库自动维护，用户无需干涉。当用户在创建 HUGE 表时，未指定 HTS 表空间的情况下，充当默认 HTS 表空间。

每一个用户都有一个默认的表空间。对于 SYS、SYSSSO、SYSAUDITOR 系统用户，默认的用户表空间是 SYSTEM，SYSDBA 的默认表空间为 MAIN，新创建的用户如果没有指定默认表空间，则系统自动指定 MAIN 表空间为用户默认的表空间。如果用户在创建表的时候，指定了存储表空间 A，并且和当前用户的默认表空间 B 不一致时，表存储在用户指定的表空间 A 中，并且默认情况下，在这张表上面建立的索引也将存储在 A 中，但是用户的默认表空间是不变的，仍为 B。

一般情况下，建议用户自己创建一个表空间来存放业务数据，或者将数据存放在默认的用户表空间 MAIN 中。

用户可以通过执行如下语句来查看表空间相关信息。

SYSTEM、ROLL、MAIN 和 TEMP 表空间查看语句：

```
SELECT * FROM V$TABLESPACE;
```

HMAIN 表空间查看语句：

```
SELECT * FROM V$HUGE_TABLESPACE;
```

4.2.2 记录

数据库表中的每一行是一条记录。在 DM 中，除了 Huge 表，其他的表都是在数据页中按记录存储数据的。也就是说，记录是存储在数据页中的，记录并不是 DM 数据库的存储单位，页才是。由于记录不能跨页存储，这样记录的长度就受到数据页大小的限制。数据页中还包含了页头控制信息等空间，因此 DM 规定每条记录的总长度不能超过页面大小的一半。

4.2.3 页

数据页（也称数据块）是 DM 数据库中最小的数据存储单元。页的大小对应物理存储空间上特定数量的存储字节，在 DM 数据库中，页大小可以为 4KB、8KB、16KB 或者 32KB，用户在创建数据库时可以指定，默认大小为 8KB，一旦创建好了数据库，则在该库的整个生命周期内，页大小都不能够改变。图 1.2 显示了 DM 数据库页的典型格式。

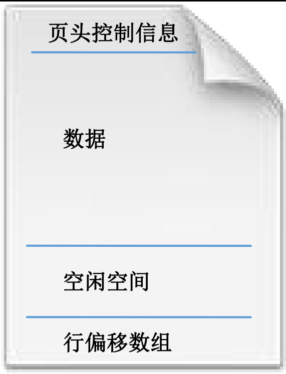


图 1.2 DM 数据页的组成

页头控制信息包含了关于页类型、页地址等信息。页的中部存放数据，为了更好地利用数据页，在数据页的尾部专门留出一部分空间用于存放行偏移数组，行偏移数组用于标识页上的空间占用情况以便管理数据页自身的空间。

在绝大多数情况下，用户都无需干预 DM 数据库对数据页的管理。但是 DM 数据库还是提供了选项供用户选择，使得在某些情况下能够为用户提供更佳的数据处理性能。

FILLFACTOR 是 DM 数据库提供的一个与性能有关的数据页级存储参数，它指定一个数据页初始化后插入数据时最大可以使用空间的百分比（100），该值在创建表/索引时可以指定。设置 FILLFACTOR 参数的值，是为了指定数据页中的可用空间百分比（FILLFACTOR）和可扩展空间百分比（100-FILLFACTOR）。可用空间用来执行更多的 INSERT 操作，可扩展空间用来为数据页保留一定的空间，以防止在今后的更新操作中增加列或者修改变长列的长度时，引起数据页的频繁分裂。当插入的数据占据的数据页空间百分比低于 FILLFACTOR 时，允许数据插入该页，否则将当前数据页中的数据分为两部分，一部分保留在当前数据页中，另一部分存入一个新页中。

对于 DBA 来说，使用 FILLFACTOR 时应该在空间和性能之间进行权衡。为了充分利用空间，用户可以设置一个很高的 FILLFACTOR 值，如 100，但是这可能会导致在后续更新数据时，频繁引起页分裂，而导致需要大量的 I/O 操作。为了提高更新数据的性能，可以设置一个相对较低（但不是过低）的 FILLFACTOR 值，使得后续执行更新操作时，可以尽量避免数据页的分裂，提升 I/O 性能，不过这是以牺牲空间利用率来换取性能的提高。

4.2.4 簇

簇是数据页的上级逻辑单元，由同一个数据文件中 16 个或 32 个连续的数据页组成。在 DM 数据库中，簇的大小由用户在创建数据库时指定，默认大小为 16。假定某个数据文件大小为 32MB，页大小为 8KB，则共有 32MB/8KB/16=256 个簇，每个簇的大小为 8K*16=128K。和数据页的大小一样，一旦创建好数据库，此后该数据库的簇的大小就不能够改变。

1. 分配数据簇

当创建一个表/索引的时候，DM 为表/索引的数据段分配至少一个簇，同时数据库会自动生成对应数量的空闲数据页，供后续操作使用。如果初始分配的簇中所有数据页都已经用完，或者新插入/更新数据需要更多的空间，DM 数据库将自动分配新的簇。在缺省情况下，DM 数据库在创建表/索引时，初始分配 1 个簇，当初始分配的空间用完时，DM 数据库会自动扩展。

当 DM 数据库的表空间为新的簇分配空闲空间时，首先在表空间按文件从小到大的顺序在各个数据文件中查找可用的空闲簇，找到后进行分配；如果各数据文件都没有空闲簇，则在各数据文件中查找空闲空间足够的，将需要的空间先进行格式化，然后进行分配；如果各文件的空闲空间也不够，则选一个数据文件进行扩充。

达梦数据库信创大赛资料

2． 释放数据簇

对于用户数据表空间，在用户将一个数据段对应的表/索引对象 DROP 之前，该表对应的数据段会保留至少 1 个簇不被回收到表空间中。在删除表/索引对象中的记录的时候，DM 数据库通过修改数据文件中的位图来释放簇，释放后的簇被视为空闲簇，可以供其他对象使用。当用户删除了表中所有记录时，DM 数据库仍然会为该表保留 1-2 个簇供后续使用。若用户使用 DROP 语句来删除表/索引对象，则此表/索引对应的段以及段中包含的簇全部收回，并供存储于此表空间的其他模式对象使用。

对于临时表空间，DM 数据库会自动释放在执行 SQL 过程中产生的临时段，并将属于此临时段的簇空间还给临时表空间。需要注意的是，临时表空间文件在磁盘所占大小并不会因此而缩减，用户可以通过系统函数 SF_RESET_TEMP_TS 来进行磁盘空间的清理。

对于回滚表空间，DM 数据库将定期检查回滚段，并确定是否需要从回滚段中释放一个或多个簇。

4.2.5 段

段是簇的上级逻辑分区单元，它由一组簇组成。在同一个表空间中，段可以包含来自不同文件的簇，即一个段可以跨越不同的文件。而一个簇以及该簇所包含的数据页则只能来自一个文件，是连续的 16 或者 32 个数据页。由于簇的数量是按需分配的，数据段中的不同簇在磁盘上不一定连续。

1． 数据段

段可以被定义成特定对象的数据结构，如表数据段或索引数据段。表中的数据以表数据段结构存储，索引中的数据以索引数据段结构存储。DM 以簇为单位给每个数据段分配空间，当数据段的簇空间用完时，DM 数据库就给该段重新分配簇，段的分配和释放完全由 DM 数据库自动完成，可以在创建表/索引时设置存储参数来决定数据段的簇如何分配。

当用户使用 CREATE 语句创建表/索引时，DM 创建相应的数据段。表/索引的存储参数用来决定对应数据段的簇如何被分配，这些参数将会影响与对象相关的数据段的存储与访问效率。对于分区表，每个分区使用单独的数据段来容纳所有数据，对于分区表上的非分区索引，使用一个索引数据段来容纳所有数据，而对于分区索引，每个分区使用一个单独索引数据段来容纳其数据。表的数据段和与其相关的索引段不一定要存储在同一表空间中，用户可以在创建表和索引时，指定不同的表空间存储参数。

2． 临时段

在 DM 数据库中，所有的临时段都创建在临时表空间中，这样可以分流磁盘设备的 I/O，也可以减少由于在 SYSTEM 或其他表空间内频繁创建临时数据段而造成的碎片。

当处理一个查询时，经常需要为 SQL 语句的解析与执行的中间结果准备临时空间。DM 数据库会自动地分配临时段的磁盘空间。例如，DM 在进行排序操作时就可能需要使用临时段，当排序操作可以在内存中执行，或设法利用索引就可以执行时，就不必创建临时段。对于临时表及其索引，DM 数据库也会为它们分配临时段。

临时段的分配和释放完全由系统自动控制，用户不能手工进行干预。

3． 回滚段

DM 数据库在回滚表空间的回滚段中保存了用于恢复数据库操作的信息。对于未提交事务，当执行回滚语句时，回滚记录被用来做回滚变更。在数据库恢复阶段，回滚记录被用来做任何未提交变更的回滚。在多个并发事务运行期间，回滚段还为用户提供读一致性，所有正在读取受影响行的用户将不会看到行中的任何变动，直到他们事务提交后发出新的查询。

DM 数据库提供了全自动回滚管理机制来管理回滚信息和回滚空间，自动回滚管理消除了

达梦数据信创大赛资料

管理回滚段的复杂性。此外，系统将尽可能保存回滚信息，来满足用户查询回滚信息的需要。事务被提交后，回滚数据不能再回滚或者恢复，但是从数据读一致性的角度出发，长时间运行查询可能需要这些早期的回滚信息来生成早期的数据页镜像，基于此，数据库需要尽可能长时间的保存回滚信息。DM 数据库会收集回滚信息的使用情况，并根据统计结果对回滚信息保存周期进行调整，数据库将回滚信息保存周期设为比系统中活动的最长的查询时间稍长。

达梦数据库信创大赛资料

第 5 章 DM 物理存储结构

DM 数据库使用了磁盘上大量的物理存储结构来保存和管理用户数据。典型的物理存储结构包括：用于进行功能设置的配置文件；用于记录文件分布的控制文件；用于保存用户实际数据的数据文件、重做日志文件、归档日志文件、备份文件；用来进行问题跟踪的跟踪日志文件等，如图 5.1 所示。

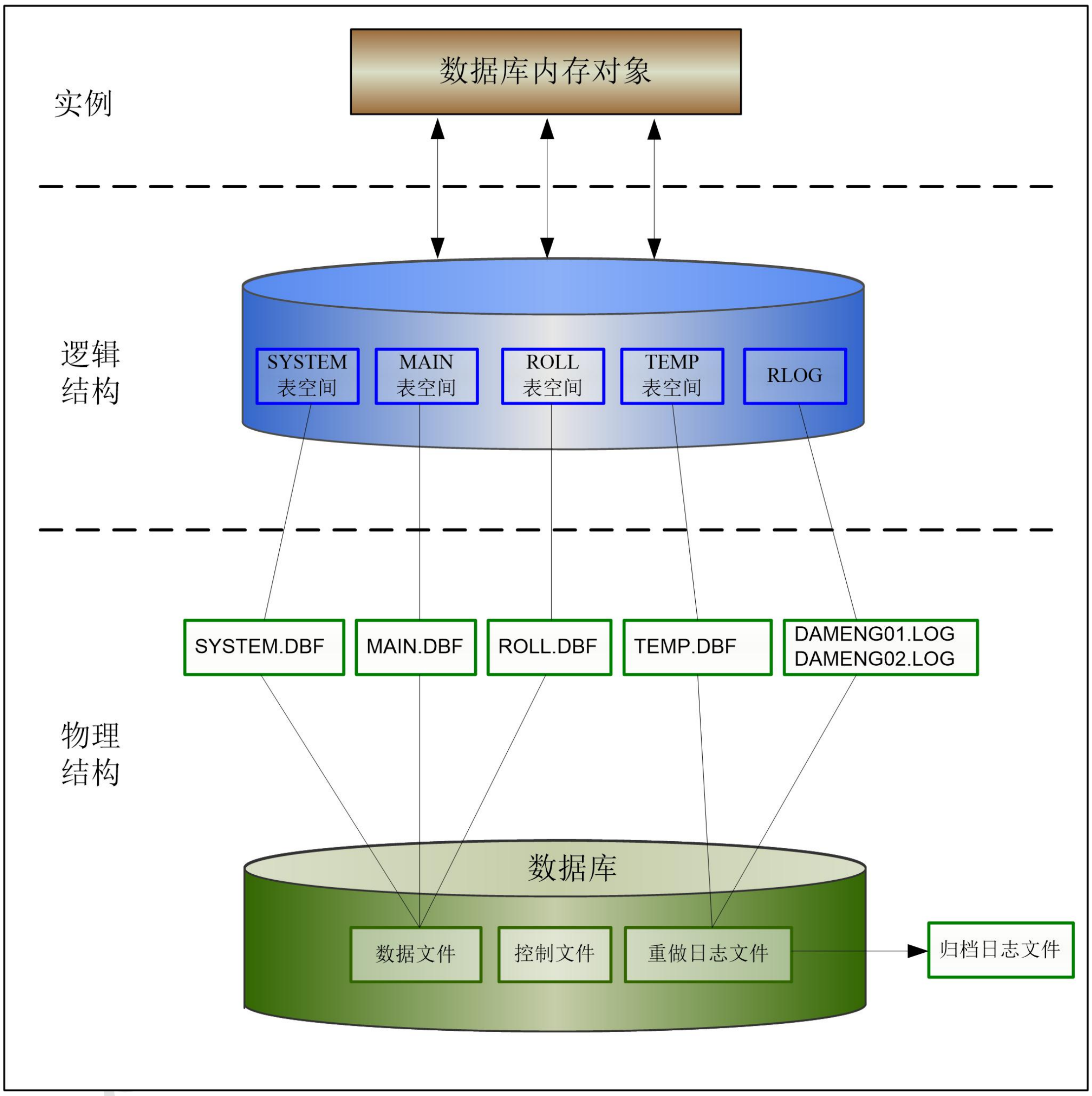


图 5.1 DM 物理存储结构示意图

5.1 配置文件

配置文件是 DM 数据库用来设置功能选项的一些文本文件的集合，配置文件以 ini 为扩展名，它们具有固定的格式，用户可以通过修改其中的某些参数取值来达成如下两个方面的目标：

1．启用/禁用特定功能项；

2．针对当前系统运行环境设置更优的参数值以提升系统性能。

5.1.1 DM 数据库服务配置

1) dm.ini

每创建一个 DM 数据库，就会自动生成 dm.ini 文件。dm.ini 是 DM 数据库启动所必须的配置文件，通过配置该文件可以设置 DM 数据库服务器的各种功能和性能选项，主要的配置内容见表 5.1。

参数属性分为三种：静态、动态和手动。

静态，可以被动态修改，修改后重启服务器才能生效。

动态，可以被动态修改，修改后即时生效。动态参数又分为会话级和系统级两种。会话级参数被修改后，新参数值只会影响新创建的会话，之前创建的会话不受影响；系统级参数的修改则会影响所有的会话。

手动，不能被动态修改，必须手动修改 dm.ini 参数文件，然后重启才能生效。

动态修改是指 DBA 用户可以在数据库服务器运行期间，通过调用系统过程 SP_SET_PARA_VALUE()、SP_SET_PARA_DOUBLE_VALUE()和 SP_SET_PARA_STRING_VALUE()对参数值进行修改。

5.2 控制文件

每个 DM 数据库都有一个名为 dm.ctl 的控制文件。控制文件是一个二进制文件，它记录了数据库必要的初始信息，其中主要包含以下内容：

1. 数据库名称；
2. 数据库服务器模式；
3. OGUID 唯一标识；
4. 数据库服务器版本；
5. 数据文件版本；
6. 数据库的启动次数；
7. 数据库最近一次启动时间；
8. 表空间信息，包括表空间名，表空间物理文件路径等，记录了所有数据库中使用的表空间，数组的方式保存起来；
9. 控制文件校验码，校验码由数据库服务器在每次修改控制文件后计算生成，保证控制文件合法性，防止文件损坏及手工修改。

在服务器运行期间，执行表空间的 DDL 等操作后，服务器内部需要同步修改控制文件内容。如果在修改过程中服务器故障，可能会导致控制文件损坏，为了避免出现这种情况，在修改控制文件时系统内部会执行备份操作。备份策略如下：

◆ 策略一 在修改 dm.ctl 之前，先执行一次备份，确定 dm.ctl 修改成功后，再将备份删除，如果 dm.ctl 修改失败或中途出现故障，则保留备份文件。

◆ 策略二 在修改 dm.ctl 成功之后，根据 dm.ini 中指定的 CTL_BAK_PATH/CTL_BAK_NUM 对最新的 dm.ctl 执行备份，如果用户指定的 CTL_BAK_PATH 是非法路径，则不再生成备份文件，在路径有效的情况下，生成备份文件时根据指定的 CTL_BAK_NUM 判断是否删除老的备份文件。

注意：

- 如果 dm.ctl 文件存放在裸设备上，则【策略一】不会生效。
- 如果指定的 CTL_BAK_PATH 是无效路径，则【策略二】也不会生效。
- 如果【策略一】和【策略二】的条件都满足，则都会生效执行，否则只执行满足条

达梦数据库信创大赛资料

件的备份策略，如果都不满足，则不会再生成备份文件。

- 如果是初始化新库，在初始化完成后，会在“SYSTEM_PATH/CTL_BAK”路径下对原始的 dm.ctl 执行一次备份。

5.3 数据文件

数据文件以 dbf 为扩展名，它是数据库中最重要的文件类型，一个 DM 数据文件对应磁盘上的一个物理文件，数据文件是真实数据存储的地方，每个数据库至少有一个与之相关的数据文件。在实际应用中，通常有多个数据文件。

当 DM 的数据文件空间用完时，它可以自动扩展。可以在创建数据文件时通过 MAXSIZE 参数限制其扩展量，当然，也可以不限制。但是，数据文件的大小最终会受物理磁盘大小的限制。在实际使用中，一般不建议使用单个巨大的数据文件，为一个表空间创建多个较小的数据文件是更好的选择。

数据文件在物理上按照页、簇和段的方式进行管理，详细结构请参考第一章的内容。

数据文件按数据组织形式，可以分为如下几种：

1. B 树数据

行存储数据，也是应用最广泛的存储形式，其数据是按 B 树索引组织的。普通表、分区表、B 树索引的物理存储格式都是 B 树。

一个 B 树包含两个段，一个内节点段，存放内节点数据；一个叶子段，存放叶子节点数据。其 B 树的逻辑关系由段内页面上的记录，通过文件指针来完成。

当表上没有指定聚簇索引时，系统会自动产生一个唯一标识 rowid 作为 B 树的 key 来唯一标识一行。

1). ROLL 文件

ROLL 表空间的 dbf 文件，称为 ROLL 文件。ROLL 文件用于保存系统的回滚记录，提供事务回滚时的信息。回滚文件整个是一个段。每个事务的回滚页在回滚段中各自挂链，页内则顺序存放回滚记录。

2). TEMP 文件

TEMP.DBF 临时数据文件，临时文件可以在 dm.ini 中通过 TEMP_SIZE 配置大小。

当数据库查询的临时结果集过大，缓存已经不够用时，临时结果集就可以保存在 TEMP.DBF 文件中，供后续运算使用。系统中用户创建的临时表也存储在临时文件中。

5.4 重做日志文件

重做日志（即 REDO 日志）指在 DM 数据库中添加、删除、修改对象，或者改变数据，DM 都会按照特定的格式，将这些操作执行的结果写入到当前的重做日志文件中。重做日志文件以 log 为扩展名。每个 DM 数据库实例必须至少有 2 个重做日志文件，默认两个日志文件为 DAMENG01.log、DAMENG02.log，这两个文件循环使用。

重做日志文件因为是数据库正在使用的日志文件，因此被称为联机日志文件。

重做日志文件主要用于数据库的备份与恢复。理想情况下，数据库系统不会用到重做日志文件中的信息。然而现实世界总是充满了各种意外，比如电源故障、系统故障、介质故障，或者数据库实例进程被强制终止等，数据库缓冲区中的数据页会来不及写入数据文件。这样，在重启 DM 实例时，通过重做日志文件中的信息，就可以将数据库的状态恢复到发生意外时的状态。

重做日志文件对于数据库是至关重要的。它们用于存储数据库的事务日志，以便系统在出现系统故障和介质故障时能够进行故障恢复。在 DM 数据库运行过程中，任何修改数据库

的操作都会产生重做日志，例如，当一条元组插入到一个表中的时候，插入的结果写入了重做日志，当删除一条元组时，删除该元组的事实也被写了进去，这样，当系统出现故障时，通过分析日志可以知道在故障发生前系统做了哪些动作，并可以重做这些动作使系统恢复到故障之前的状态。

5.5 归档日志文件

日志文件分为联机日志文件和归档日志文件。DM 数据库可以在归档模式和非归档模式下运行。非归档模式下，数据库会只将重做日志写入联机日志文件中进行存储；归档模式下，数据库会同时将重做日志写入联机日志文件和归档日志文件中分别进行存储。

联机日志文件指的是系统当前正在使用的日志文件。创建数据库时，联机日志文件通常被扩展至一定长度，其内容则被初始化为空，当系统运行时，该文件逐渐被产生的日志所填充。对日志文件的写入是顺序连续的。然而系统磁盘空间总是有限，系统必须能够循环利用日志文件的空间，为了做到这一点，当所有日志文件空间被占满时，系统需要清空一部分日志以便重用日志文件的空间，为了保证被清空的日志所“保护”的数据在磁盘上是安全的，这里需要引入一个关键的数据库概念—检查点。当产生检查点时，系统将系统缓冲区中的日志和脏数据页都写入磁盘，以保证当前日志所“保护”的数据页都已安全写入磁盘，这样日志文件即可被安全重用。

归档日志文件，就是在归档模式下，重做日志被连续写入到归档日志后，所生成了归档日志文件。归档日志文件以归档时间命名，扩展名也是 log。但只有在归档模式下运行时，DM 数据库才会将重做日志写入到归档日志文件中。采用归档模式会对系统的性能产生影响，然而系统在归档模式下运行会更安全，当出现故障时其丢失数据的可能性更小，这是因为一旦出现介质故障，如磁盘损坏时，利用归档日志，系统可被恢复至故障发生的前一刻，也可以还原到指定的时间点，而如果没有归档日志文件，则只能利用备份进行恢复。归档日志文件还是数据守护功能的核心，数据守护中的备库就是通过重做归档日志中的 REDO 日志来完成与主库的数据同步的。

5.6 备份文件

备份文件以 bak 为扩展名，当系统正常运行时，备份文件不会起任何作用，它也不是数据库必须有的联机文件类型之一。然而，从来没有哪个数据库系统能够保证永远正确无误地运行，当数据库不幸出现故障时，备份文件就显得尤为重要了。

当客户利用管理工具或直接发出备份的 SQL 命令时，DM Server 会自动进行备份，并产生一个或多个备份文件，备份文件自身包含了备份的名称、对应的数据库、备份类型和备份时间等信息。同时，系统还会自动记录备份信息及该备份文件所处的位置，但这种记录是松散的，用户可根据需要将其拷贝至任何地方，并不会影响系统的运行。

第 6 章 管理表

表是数据库中数据存储的基本单元，是对用户数据进行读和操纵的逻辑实体。表由列和行组成，每一行代表一个单独的记录。表中包含一组固定的列，表中的列描述该表所跟踪的实体的属性，每个列都有一个名字及各自的特性。

列的特性由两部分组成：数据类型（dataType）和长度（length）。对于 NUMERIC、DECIMAL 以及那些包含秒的时间间隔类型来说，可以指定列的小数位及精度特性。在 DM 系

统中，CHAR、CHARACTER、VARCHAR 数据类型的最大长度由数据库页面大小决定，数据库页面大小在初始化数据库时指定。DM 系统具有 SQL-92 的绝大部分数据类型，以及部分 SQL-99、Oracle 和 SQL Server 的数据类型。

为了确保数据库中数据的一致性和完整性，在创建表时可以定义表的实体完整性、域完整性和参考完整性。实体完整性定义表中的所有行能唯一地标识，一般用主键、唯一索引、UNIQUE 关键字及 IDENTITY 属性来定义；域完整性通常指数据的有效性，限制数据类型、缺省值、规则、约束、是否可以为空等条件，域完整性可以确保不会输入无效的值；参考完整性维护表间数据的有效性、完整性，通常通过建立外键联系另一表的主键来实现。

如果用户在创建表时没有定义表的完整性和一致性约束条件，那么用户可以利用 DM 所提供的基表修改语句来进行补充或修改。DM 系统提供基表修改语句，可对基表的结构进行全面的修改，包括修改基表名、列名、增加列、删除列、修改列类型、增加表级约束、删除表级约束、设置列缺省值、设置触发器状态等一系列修改功能。

本章描述管理表的几个方面，包括以下内容：

1. 管理表的准则；
2. 创建表；
3. 更改表；
4. 删除表；
5. 清空表；
6. 查看表信息。

6.1 管理表的准则

6.1.1 设计表

表是数据库设计过程中的基本构件，基于来自应用开发者的有关应用如何运作和所期望的数据类型，数据库管理员应与应用开发者一起工作，并认真规划每个表，具体需要做到以下几点：

1. 规范化表，估算并校正表结构，使数据冗余达到最小；
2. 为每个列选择合适的数据类型，是否允许为空等，并根据实际情况判断是否需要对列进行加密或压缩处理；
3. 建立合适的完整性约束；
4. 建立合适的聚集索引。每个表（列存储表，堆表除外）都含一个聚集索引，默认以 ROWID 建立，而建立合适的聚集索引，可以有效加快表的检索效率；
5. 根据实际需要，建立合适类型的表。DM 支持的表类型包括普通表、临时表、水平分区表、堆表和列存储表。本章只介绍普通表和临时表，其他类型表将在其他章节中重点介绍。

6.1.2 指定表的存储空间上限

在创建表时指定 SPACE LIMIT 子句，可以对表的存储空间指定上限。DM 支持对表的存储空间指定大小，单位是 MB，即表的大小可由管理员指定，便于表的规模管理。当表的所有索引所占用的存储空间超过指定大小时，表将不能再新增数据。

6.1.3 指定表的存储位置

创建表时，在 STORAGE 子句中，可对表指定存储的表空间。如果没有指定，则该表将创建在用户的默认表空间中。

在创建表时，通过指定合适的表空间，有以下优点：

1. 提高数据库系统的性能，因为不同的数据库表可能对应不同的数据文件，可减少对相同文件的竞争；
2. 减少数据库管理的时间，数据库表分布在不同的表空间中，即使一个表空间损坏，也不影响其他表空间上数据库表的正常访问。

6.2 创建表

6.2.1 创建普通表

如果要在所属模式中创建新表，需要有 CREATE TABLE 数据库权限；而要在其他用户的模式中创建新表，则需要有 CREATE ANY TABLE 数据库权限。创建表时，应当为表指定一个表空间，否则，表将在 MAIN 创建。下面给出一个创建一个简单表的例子。

```
CREATE  TABLE  EMPLOYEE (
  EMPNO  INT              PRIMARY KEY,
  ENAME  VARCHAR(15)      NOT NULL,
  JOB        VARCHAR(10),
  MGR        INT
      CONSTRAINT EMP_FKEY REFERENCES EMPLOYEE(EMPNO),
  HIREDATE   DATE             DEFAULT (CURDATE),
  SALARY         FLOAT,
  DEPTNO     TINYINT          NOT NULL
CONSTRAINT DEPT_FKEY REFERENCES DEPT(DEPTNO))
STORAGE (
INITIAL         50,
NEXT            50,
MINEXTENTS      10,
FILLFACTOR      80,
ON              USERS);
```

在上述 CREATE TABLE 语句中，在 users 表空间上建立了 employee 表，并有几个完整性约束，其中包含定义在不同列上的一个主键和外键。约束将在 15 章中讨论。

创建表之后，可以使用 INSERT 命令插入数据或使用达梦数据导入导出工具装载数据，还可以直接使用 CREATE TABLE AS SELECT 创建一个表。

6.2.2 指定表的聚集索引

表（列存储表和堆表除外）都是使用 B+树(以下简称 B 树)索引结构管理的，每一个普通表都有一个聚集索引，数据通过聚集索引键排序，根据聚集索引键可以快速查询任何记录。

当建表语句未指定聚集索引键，DM 的默认聚集索引键是 ROWID，即记录默认以 ROWID 在页面中排序。ROWID 是 B 树为记录生成的逻辑递增序号，表上不同记录的 ROWID 是不一

样的，并且最新插入的记录 ROWID 最大。很多情况下，以 ROWID 建的默认聚集索引并不能提高查询速度，因为实际情况下很少人根据 ROWID 来查找数据。

因此，DM 提供三种方式供用户指定聚集索引键：

1. CLUSTER PRIMARY KEY：指定列为聚集索引键，并同时指定为主键，称为聚簇主键；
2. CLUSTER KEY：指定列为聚集索引键，但是是非唯一的；
3. CLUSTER UNIQUE KEY：指定列为聚集索引键，并且是唯一的。

例如，创建 student 表，指定 stu_no 为聚簇主键。

```
CREATE  TABLE  STUDENT(
  STUNO      INT             CLUSTER PRIMARY KEY,
  STUNAME    VARCHAR(15)     NOT NULL,
  TEANO      INT,
  CLASSID    INT
);
```

指定聚簇索引键后，如果查询条件中含有聚簇索引键，可以定位记录在 B 树上的位置，使查询性能大大提高。然而，插入记录也需要根据聚簇索引键定位插入位置，有可能导致页面的分裂而影响插入性能。

在 dm.ini 配置文件中，可以指定配置项使表中的主键自动转化为聚簇主键，该配置项为 PK_WITH_CLUSTER。默认情况下，PK_WITH_CLUSTER 为 1，即建表时指定的主键自动转化为聚簇主键；若为 0，则主键不会自动变为聚簇主键。

6.2.3 查询建表

为了创建一个与已有表相同的新表，或者为了创建一个只包含另一个表的一些行和列的新表，可以使用 CREATE TABLE AS SELECT(CTAS) 命令。使用该命令，可以通过使用 WHERE 条件将已有表中的一部分数据装载到一个新表中，或者可以通过 SELECT * FROM 子句将已有表的所有数据装载到创建的表中，如以下程序片段所示：

```
CREATE TABLE NEW_EMP
AS
SELECT * FROM EMPLOYEE;
```

如果用户通过单表的全表查询进行建表操作，则可以通过将 INI 参数 CTAB_SEL_WITH_CONS 置为 1 进行原始表上约束的拷贝，列上能拷贝的约束包括默认值属性、自增属性、非空属性以及加密属性，表上能拷贝的约束包括唯一约束、PK 约束以及 CHECK 约束。

6.3 更改表

想更改的表如果在所属的模式中，用户必须具有 ALTER TABLE 数据库权限；若在其他模式中，用户必须有 ALTER ANY TABLE 的数据库权限。

通过更改表，用户可以对数据库中的表作如下修改：

1. 添加或删除列，或修改现有的列的定义（列名、数据类型、长度、默认值）。其中，对于添加列，当设置 INI 参数 ALTER_TABLE_OPT 为 1 时，添加列采用查询插入实现，可能会导致 ROWID 的改变；ALTER_TABLE_OPT 为 2 时，系统开启快速加列功能，对于没有默认值或者默认值为 NULL 的新列，系统内部会标记为附加列，能够达到瞬间加列的效果，此时 ROWID 不会改变，若有默认值且默认值不为 NULL，则仍旧采取查询插入实现。

2． 添加、修改或删除与表相关的完整性约束；
3． 重命名一个表；
4． 启动或停用与表相关的完整性约束；
5． 启动或停用与表相关的触发器；
6． 修改表的 SPACE LIMIT；
7． 增删自增列。

6.4 删除表

当一个表不再使用时，可以将其删除。删除表时，将产生以下结果：
1． 表的结构信息从数据字典中删除，表中的数据不可访问；
2． 表上的所有索引和触发器被一起清除；
3． 所有建立在该表上的同义词、视图和存储过程变为无效；
4． 所有分配给表的簇标记为空闲，可被分配给其他的数据库对象。

一般情况下，普通用户只能删除自己模式下的表。若要删除其他模式下的表，则必须具有 DROP ANY TABLE 数据库权限。

以下语句可删除 employee 表：

```
DROP TABLE employee;
```

如果要删除的表被其他表引用，即其他表的外键引用了表的任何主键或唯一键，则需要在 DROP TABLE 语句中包含 CASCADE 选项，如：

```
DROP TABLE employee CASCADE;
```

6.5 清空表

有些情况下，当表的数据不再使用时，需要删除表的所有行，即清空该表。DM8 支持以下方式来删除表中的所有的行：
1． 使用 DELETE 语句；
2． 使用 DROP 和 CREATE 语句；
3． 使用 TRUNCATE 语句。

6.5.1 使用 DELETE

使用 DELETE 语句能删除表中的行。例如，下面的语句删除 employee 表中的所有行：

```
DELETE FROM employee;
```

但是，使用 DELETE 清空表，当表有很多行时，会消耗很多系统资源。因为，DELETE 操作需要 CPU 时间，并且会产生大量的 REDO 日志和 UNDO 记录。另外，如果表上关联了元组级触发器，每删除一行，就会启动一次触发器。这都需要大量的系统资源。

6.5.2 使用 DROP 和 CREATE

使用 DROP 删除一个表，然后创建一个同名的表，也可以达到清空表的效果。例如，下面的语句先删除 employe 表，之后再重新创建。

```
DROP TABLE EMPLOYEE;
CREATE TABLE EMPLOYEE(…);
```

当删除和重新创建表时，所有与之相关联的索引、完整性约束和触发器也被删除。同样，所有针对被删除表的授权也会被删除。

6.5.3 使用 TRUNCATE

使用 TRUNCATE 语句能删除表中的所有行。例如，下面的语句清空 employee 表。

```
TRUNCATE TABLE EMPLOYEE;
```

TRUNCATE 语句为我们提供了一种快速、有效地删除表所有行的方法。并且 TRUNCATE 是一个 DDL 语句，不会产生任何回滚信息。执行 TRUNCATE 会立即提交，而且不能回滚。

TRUNCATE 语句并不影响与被删除的表相关联的任何结构、约束、触发器或者授权。 另外，DM 数据库 TRUNCATE 表后，原来分配给该表的空间会被释放，供其他数据库对象使用，大大提高空间的利用效率。

一般情况下，普通用户只能 TRUNCATE 自己模式下的表。若要 TRUNCATE 其他模式下的表，则必须具有 DROP ANY TABLE 数据库权限。

如果要清空的表被其他表引用，即其他表的外键引用了表的任何主键或唯一键，并且子表不为空或子表的外键约束未被禁用，则不能 TRUNCATE 该表。

6.6 查看表信息

6.6.1 查看表定义

创建表后，可以通过 SP_TABLEDEF 系统过程查看表的定义。

```
CALL SP_TABLEDEF('SYSDBA', 'EMPLOYEE');
```

DM 通过提供的 TABLEDEF 函数来显示当前表的定义。当表多次进行 ALTER TABLE 后，显示的表定义将是最后一次修改后的建表语句。

6.6.2 查看表的空间使用情况

DM 使用段、簇和页实现数据的物理组织。DM 支持查看表的空间使用情况，包括：

1. TABLE_USED_SPACE：已分配给表的页面数；
2. TABLE_USED_PAGES：表已使用的页面数。

```
CREATE TABLE SPACE_TABLE (
    C1          INT,
    C2          INT
);
SELECT TABLE_USED_SPACE('SYSDBA','SPACE_TABLE');
SELECT TABLE_USED_PAGES('SYSDBA','SPACE_TABLE');
```

第 7 章　管理索引

7.1　创建索引

本节描述如何创建索引。要在用户自己的模式中创建索引，至少要满足如下条件之一：

1. 要被索引的表是在自己的模式中 ；
2. 在要被索引的表上有 CREATE INDEX 权限；
3. 具有 CREATE ANY INDEX 数据库权限。

要在其他模式中创建索引，用户必须具有 CREATE ANY INDEX 数据库权限。

7.1.1 明确地创建索引

可以用 CREATE INDEX 语句明确地创建索引。如下语句在 emp 表的 ename 列上创建一个名为 emp_ename 的索引，该索引使用表空间 users。

```
CREATE INDEX emp_ename ON emp(ename)
STORAGE (
INITIAL 50,
NEXT    50,
ON      USERS);
```

注意，上述语句为该索引明确地指定了几个存储设置和一个表空间。如果没有给索引指定存储选项，则 INITIAL 和 NEXT 等存储选项会自动使用表空间的默认存储选项。

7.1.2 创建聚集索引

DM8 中表（列存储表和堆表除外）都是使用 B+树索引结构管理的，每一个普通表都有且仅有一个聚集索引，数据都通过聚集索引键排序，根据聚集索引键可以快速查询任何记录。

当建表语句未指定聚集索引键时，DM8 的默认聚集索引键是 ROWID。若指定索引键，表中数据都会根据指定索引键排序。

建表后，DM8 也可以用创建新聚集索引的方式来重建表数据，并按新的聚集索引排序。例如，可以对 emp 表以 ename 列新建聚集索引。

```
CREATE CLUSTER INDEX clu_emp_name ON emp(ename);
```

新建聚集索引会重建这个表以及其所有索引，包括二级索引、函数索引，是一个代价非常大的操作。因此，最好在建表时就确定聚集索引键，或在表中数据比较少时新建聚集索引，而尽量不要对数据量非常大的表建立聚集索引。

创建聚集索引的约束条件：

1. 每张表中只允许有一个聚集索引，如果之前已经指定过 CLUSTER INDEX 或者指定了 CLUSTER PK，则用户新建立 CLUSTER INDEX 时系统会自动删除原先的聚集索引。但如果新建聚集索引时指定的创建方式（列，顺序）和之前的聚集索引一样，则会报错；
2. 指定 CLUSTER INDEX 操作需要重建表上的所有索引，包括 PK 索引；
3. 删除聚集索引时，缺省以 ROWID 排序，自动重建所有索引；

4. 若聚集索引是默认的 ROWID 索引，不允许删除；
5. 聚集索引不能应用到函数索引中；
6. 不能在列存储表上新建/删除聚集索引；
7. 建聚集索引语句不能含有 partition_clause 子句；
8. 在临时表上增删索引会使当前会话上临时 b 树数据丢失。

7.1.3 明确地创建唯一索引

索引可以是唯一的或非唯一的。唯一索引可以保证表上不会有两行数据在键列上具有相同的值，非唯一索引不会在键列上施加这个限制。

可用 CREATE UNIQUE INDEX 语句来创建唯一索引，如下例子创建一个唯一索引：

```
CREATE UNIQUE INDEX dept_unique_index ON dept (dname)
STORAGE (ON users);
```

用户可以在希望的列上定义 UNIQUE 完整性约束，DM8 通过自动地在唯一键上定义一个唯一索引来保证 UNIQUE 完整性约束。

7.1.4 自动创建与约束相关的唯一索引

DM8 通过在唯一键或主键上创建一个唯一索引来在表上实施 UNIQUE KEY 或 PRIMARY KEY 完整性约束。当启用约束时 DM8 自动创建该索引。如下面的语句会自动在表 emp 的 name 列上创建一个唯一索引。

```
ALTER TABLE EMP ADD CONSTRAINT PK_EMP_NAME PRIMARY KEY (NAME);
```

7.2 重建索引

当一个表经过大量的增删改操作后，表的数据在物理文件中可能存在大量碎片，从而影响访问速度。另外，当删除表的大量数据后，若不再对表执行插入操作，索引所处的段可能占用了大量并不使用的簇，从而浪费了存储空间。

可以使用重建索引来对索引的数据进行重组，使数据更加紧凑，并释放不需要的空间，从而提高访问效率和空间效率。DM8 提供的重建索引的系统函数为：

```
SP_REBUILD_INDEX(SCHEMA_NAME varchar(256), INDEX_ID int);
```

SCHEAM_NAME 为索引所在的模式名，INDEX_ID 为索引 ID。

使用说明：

1. 水平分区子表，临时表和系统表上建的索引不支持重建
2. 虚索引和聚集索引不支持重建

例如，需要重建索引 emp_name，假设其索引 ID 为 1547892，那么使用以下语句重建索引：

```
SP_REBUILD_INDEX('SYSDBA', 1547892);
```

7.3 删除索引

用户可能出于以下某项原因需要删除一个索引：

1. 不再需要该索引；
2. 该索引没有为针对其相关的表所发布的查询提供所期望的性能改善。例如，表可能

很小，或者尽管表中有许多行但只有很少的索引项；

3. 应用没有用该索引来查询数据。

要想删除索引，则该索引必须包含在用户的模式中或用户必须具有 DROP ANY INDEX 数据库权限。索引删除之后，该索引的段的所有簇都返回给包含它的表空间，并可用于表空间中的其他对象。

如何删除索引，取决于是否是用 CREATE INDEX 语句明确地创建该索引的，是则可以用 DROP INDEX 语句删除该索引。如下面的语句删除 emp_ename 索引。

```
DROP INDEX emp_ename;
```

然而，不能直接删除与已启用的 UNIQUE KEY 键或 PRIMARY KEY 键约束相关的索引。要删除一个与约束相关的索引，必须停用或删除该约束本身。如下面的语句删除主键约束 pk_emp_name，同时删除其对应的索引。

```
ALTER TABLE emp DROP CONSTRAINT pk_emp_name;
```

除了删除普通索引，DM8 还提供删除聚集索引，只要其聚集索引是通过 CREATE CLUSTER INDEX 明确建立的。例如，下面的语句删除 emp 表的聚集索引 clu_emp_name。

```
DROP INDEX clu_emp_name;
```

删除聚集索引其实是使用 ROWID 作为索引列重建聚集索引，即跟新建聚集索引一样会重建这个表以及其所有索引。

删除表就自动删除了所有与其相关的索引。

7.4 查看索引信息

创建索引后，可以通过 INDEXDEF 系统函数查看索引的定义。

```
INDEXDEF(INDEX_ID int, PREFLAG int);
```

INDEX_ID 为索引 ID，PREFLAG 表示返回信息中是否增加模式名前缀。例如，需要查看索引 emp_name 的定义，假设其索引 ID 为 1547892，那么使用以下语句查看索引定义。

```
SELECT INDEXDEF(1547892, 0);
或 SELECT INDEXDEF(1547892, 1);
```

达梦数据库创大赛资料

第 8 章 管理视图

8.1 管理视图

视图是关系数据库系统提供给用户以多种角度观察数据库中数据的重要机制，它简化了用户数据模型，提供了逻辑数据独立性，实现了数据共享和数据的安全保密。视图是数据库技术中一个十分重要的功能。从系统实现的角度讲，视图是从一个或几个基表（或视图）导出的表，但它是一个虚表，即数据字典中只存放视图的定义（由视图名和查询语句组成），而不存放对应的数据，这些数据仍存放在原来的基表中。当对一个视图进行查询时，视图将查询其对应的基表，并且将所查询的结果以视图所规定的格式和次序进行返回。因此当基表中的数据发生变化时，从视图中查询出的数据也随之改变了。从用户的角度来讲，视图就像一个窗口，透过它可以看到数据库中用户感兴趣的数据和变化。当用户所需的数据是一张表的部分列、或部分行，或者数据是分散在多个表中，那么就可以创建视图来将这些满足条件的行和列组织到一个表，而不需要修改表的属性、甚至创建新的表。这样不仅简化了用户的操作，还可以提高数据的逻辑独立性，实现数据的共享和保密。

例如，为了防止非公司人员查询供应商的具体信息，包括 email、phone 等，这时可以建立视图，针对不用的用户建立查询供应商信息的视图。

```
CREATE VIEW normal_view AS SELECT name FROM person;
CREATE VIEW special_view AS SELECT name, sex, email, phone FROM person;
```

然后把视图 normal_view 的权限授予普通用户，只允许他们查询供应商的姓名信息，把 special_view 视图的权限授予高级用户，允许他们查看供应商的 email、电话等信息，这样就较好地实现了供应商信息的安全与保密。

关于视图的定义、删除、查询和修改可以参考《DM8_SQL 语言使用手册》。

用户也可以对那些经常进行的查询建立相应视图，如果经常使用的查询语句较庞大，这样可以节省繁琐语句的书写。

由于视图没有直接相关的物理数据，所以不能被索引。

DM 提供了一些以"V$"开头的视图来供用户了解当前服务器的使用情况。这些视图的数据在服务器运行期间一直变化，并且数据反映的主要是系统的性能信息，因此它们被称为动态性能视图。用户对动态性能视图只能进行查询操作，动态视图的具体内容可参考附录 2。

8.2 查看视图

视图、序列以及同义词的定义信息可以通过查看系统表 SYSOBJECTS 和 SYSTEXTS 得到，如查看普通视图 view_1 的信息，可以执行：

```
SELECT b.* FROM  SYS.SYSOBJECTS a, SYS.SYSTEXTS b WHERE a.ID = b.ID and a.NAME LIKE 'VIEW_1%';
```

可以得到：

```
ID            SEQNO         TXT
1        16778216         0          CREATE VIEW VIEW_1 AS SELECT ID FROM SYSOBJECTS;
2      16778216       1        SELECT ID FROM SYSOBJECTS
```

这样可以很明确知道 VIEW_1 视图的定义。

视图的定义信息也可以通过 SP_VIEWDEF 系统过程来查看。对于物化视图而言，其定义

只有通过该系统函数才能完整地获取。

```
CALL SP_VIEWDEF('SYSDBA', 'VIEW1');
```

达梦数据库信创大赛资料

第 9 章　结构化查询语言 DM_SQL 简介

结构化查询语言 SQL(Structured Query Language)是在 1974 年提出的一种关系数据库语言。由于 SQL 语言接近英语的语句结构，方便简洁、使用灵活、功能强大，倍受用户及计算机工业界的欢迎，被众多计算机公司和数据库厂商所采用，经各公司的不断修改、扩充和完善，SQL 语言最终发展成为关系数据库的标准语言。

SQL 的第一个标准是 1986 年 10 月由美国国家标准化组织(ANSI)公布的 ANSI X3.135-1986 数据库语言 SQL，简称 SQL-86，1987 年国际标准化组织(ISO)也通过了这一标准。以后通过对 SQL-86 的不断修改和完善，于 1989 年第二次公布了 SQL 标准 ISO/IEC 9075-1989(E)，即 SQL-89。1992 年又公布了 SQL 标准 ISO/IEC 9075:1992，即 SQL-92。1999 年公布了 ISO/IEC 9075:1999，即 SQL-3(也称 SQL-99)。之后在 2003 年公布了 ISO/IEC 9075:2003，即 SQL:2003；在 2008 年公布了 ISO/IEC 9075:2008，即 SQL:2008；在 2011 年公布了 ISO/IEC 9075:2011，即 SQL:2011。由于 SQL 标准的内容越来越庞杂，绝大多数情况下，说起 SQL 符合程度，其实是指 SQL-92 中最核心的部分，从 SQL-99 后不再对标准符合程度进行分级，而是改成了核心兼容性和特性兼容性。

SQL 成为国际标准以后，其影响远远超出了数据库领域。例如在 CAD、软件工程、人工智能、分布式等领域，人们不仅把 SQL 作为检索数据的语言规范，而且也把 SQL 作为检索图形、图象、声音、文字等信息类型的语言规范。目前，世界上大型的著名数据库管理系统均支持 SQL 语言，如 Oracle、Sybase、SQL Server、DB2 等。在未来相当长的时间里，SQL 仍将是数据库领域以至信息领域中数据处理的主流语言之一。

由于不同的 DBMS 产品，大都按自己产品的特点对 SQL 语言进行了扩充，很难完全符合 SQL 标准。目前在 DBMS 市场上已将 SQL 的符合率作为衡量产品质量的重要指标，并研制成专门的测试软件，如 NIST。目前，DM 数据库管理系统 SQL-92 入门级符合率达到 100%，过渡级符合率达到 95%，并且部分支持 SQL-99、SQL:2003、SQL:2008 和 SQL:2011 的特性。同时 DM 还兼容 Oracle 11g 和 SQL Server 2008 的部分语言特性。本章主要介绍 DM 系统所支持的 SQL 语言——DM_SQL 语言。

9.1 DM_SQL 语言的特点

DM_SQL 语言符合结构化查询语言 SQL 标准，是标准 SQL 的扩充。它集数据定义、数据查询、数据操纵和数据控制于一体，是一种统一的、综合的关系数据库语言。它功能强大，使用简单方便、容易为用户掌握。DM_SQL 语言具有如下特点：

1. 功能一体化

DM_SQL 的功能一体化表现在以下两个方面：

1) DM_SQL 支持多媒体数据类型，用户在建表时可直接使用。DM 系统在处理常规数据与多媒体数据时达到了四个一体化：一体化定义、一体化存储、一体化检索、一体化处理，最大限度地提高了数据库管理系统处理多媒体的能力和速度；
2) DM_SQL 语言集数据库的定义、查询、更新、控制、维护、恢复、安全等一系列操作于一体，每一项操作都只需一种操作符表示，格式规范，风格一致，简单方便，很容易为用户所掌握。

2． 两种用户接口使用统一语法结构的语言

DM_SQL 语言既是自含式语言，又是嵌入式语言。作为自含式语言，它能独立运行于联机交互方式。作为嵌入式语言，DM_SQL 语句能够嵌入到 C 和 C++语言程序中，将高级语言(也称主语言)灵活的表达能力、强大的计算功能与 DM_SQL 语言的数据处理功能相结合，完成各种复杂的事务处理。而在这两种不同的使用方式中，DM_SQL 语言的语法结构是一致的，从而为用户使用提供了极大的方便性和灵活性。

3． 高度非过程化

DM_SQL 语言是一种非过程化语言。用户只需指出“做什么”，而不需指出“怎么做”，对数据存取路径的选择以及 DM_SQL 语句功能的实现均由系统自动完成，与用户编制的应用程序与具体的机器及关系 DBMS 的实现细节无关，从而方便了用户，提高了应用程序的开发效率，也增强了数据独立性和应用系统的可移植性。

4． 面向集合的操作方式

DM_SQL 语言采用了集合操作方式。不仅查询结果可以是元组的集合，而且一次插入、删除、修改操作的对象也可以是元组的集合，相对于面向记录的数据库语言(一次只能操作一条记录)来说，DM_SQL 语言的使用简化了用户的处理，提高了应用程序的运行效率。

5． 语言简洁，方便易学

DM_SQL 语言功能强大，格式规范，表达简洁，接近英语的语法结构，容易为用户所掌握。

9.2 保留字与标识符

标识符的语法规则兼容标准 GJB 1382A-9X，标识符分为正规标识符和定界标识符两大类。

正规标识符以字母、_、$、#或汉字开头，后面可以跟随字母、数字、_、$、#或者汉字，正规标识符的最大长度是 128 个英文字符或 64 个汉字。正规标识符不能是保留字。

正规标识符的例子：A，test1，_TABLE_B，表 1。

定界标识符的标识符体用双引号括起来时，标识符体可以包含任意字符，特别地，其中使用连续两个双引号转义为一个双引号。

定界标识符的例子："table"，"A"，"!@#$"。

保留字的清单参见附录 1。

9.3 DM_SQL 语言的功能及语句

DM_SQL 语言是一种介于关系代数与关系演算之间的语言，其功能主要包括数据定义、查询、操纵和控制四个方面，通过各种不同的 SQL 语句来实现。按照所实现的功能，DM_SQL 语句分为以下几种：

1. 用户、模式、基表、视图、索引、序列、全文索引、存储过程、触发器等数据库对象的定义和删除语句，数据库、用户、基表、视图、索引、全文索引等数据库对象的修改语句；
2. 查询、插入、删除、修改语句；
3. 数据库安全语句。包括创建角色语句、删除角色语句，授权语句、回收权限语句，修改登录

口令语句，审计设置语句、取消审计设置语句等。

在嵌入方式中，为了协调 DM_SQL 语言与主语言不同的数据处理方式，DM_SQL 语言引入了游标的概念。因此在嵌入方式下，除了数据查询语句(一次查询一条记录)外，还有几种与游标有关的语句：

1． 游标的定义、打开、关闭、拨动语句；

2． 游标定位方式的数据修改与删除语句。

为了有效维护数据库的完整性和一致性，支持 DBMS 的并发控制机制，DM_SQL 语言提供了事务的回滚(ROLLBACK)与提交(COMMIT)语句。同时 DM 允许选择实施事务级读一致性，它保证同一事务内的可重复读，为此 DM 提供用户多种手动上锁语句，和设置事务隔离级别语句。

9.4 DM_SQL 所支持的数据类型

数据类型是可表示值的集。值的逻辑表示是<字值>。值的物理表示依赖于实现。DM 系统具有 SQL-92 的绝大部分数据类型，以及部分 SQL-99 和 SQL Server 2000 的数据类型。

9.4.1 常规数据类型

1． 字符数据类型

CHAR 类型

语法：CHAR[(长度)]

功能：CHAR 数据类型指定定长字符串。在基表中，定义 CHAR 类型的列时，其最大存储长度由数据库页面大小决定，可以指定一个不超过其最大存储长度的正整数作为字符长度，例如：CHAR(100)。如果未指定长度，缺省为 1。DM 确保存储在该列的所有值都具有这一长度。CHAR 数据类型最大存储长度和页面大小的对应关系请见下表 1.4.1。但是，在表达式计算中，该类型的长度上限不受页面大小限制，为 32767。DM 支持按字节存放字符串。

表 1.4.1

数据库页面大小	实际最大长度
4K	1900
8K	3900
16K	8000
32K	8188

这个限制长度只针对建表的情况，在定义变量的时候，可以不受这个限制长度的限制。另外，实际插入表中的列长度要受到记录长度的约束，每条记录总长度不能大于页面大小的一半。

CHARACTER 类型

语法：CHARACTER[(长度)]

功能：与 CHAR 相同。

VARCHAR 类型/VARCHAR2 类型

语法：VARCHAR[(长度)]

达梦数据库信创大赛资料

功能：VARCHAR 数据类型指定变长字符串，用法类似 CHAR 数据类型，可以指定一个不超过 8188 的正整数作为字符长度，例如：VARCHAR (100)。如果未指定长度，缺省为 8188。

在基表中，当没有指定 USING LONG ROW 存储选项时，插入 VARCHAR 数据类型的实际最大存储长度由数据库页面大小决定，具体最大长度算法如表 1.4.1；如果指定了 USING LONG ROW 存储选项，则插入 VARCHAR 数据类型的长度不受数据库页面大小限制。VARCHAR 类型在表达式计算中的长度上限不受页面大小限制，为 32767；

CHAR 同 VARCHAR 的区别在于前者长度不足时，系统自动填充空格，而后者只占用实际的字节空间。另外，实际插入表中的列长度要受到记录长度的约束，每条记录总长度不能大于页面大小的一半。

VARCHAR2 类型和 VARCHAR 类型用法相同。

2． 数值数据类型

1． 精确数值数据类型

NUMERIC 类型

语法：NUMERIC[(精度 [, 标度])]

功能：NUMERIC数据类型用于存储零、正负定点数。其中：精度是一个无符号整数，定义了总的数字数，精度范围是1至38。标度定义了小数点右边的数字位数。一个数的标度不应大于其精度，如果实际标度大于指定标度，那么超出标度的位数将会四舍五入省去。例如：NUMERIC(4,1)定义了小数点前面3位和小数点后面1位，共4位的数字，范围在-999.9到999.9。所有NUMERIC数据类型，如果其值超过精度，DM会返回一个出错信息，如果超过标度，则多余的位会被截断。

如果不指定精度和标度，缺省精度为38，标度无限定。

DECIMAL类型

语法：DECIMAL[(精度 [, 标度])]

功能：与 NUMERIC 相似。

DEC 类型

语法：DEC[(精度[, 标度])]

功能：与 DECIMAL 相同。

NUMBER 类型

语法：NUMBER[(精度[, 标度])]

功能：与 NUMERIC 相同。

INTEGER 类型

语法：INTEGER

功能：用于存储有符号整数，精度为 10，标度为 0。取值范围为：-2147483648 (-2^{31})～ +2147483647($2^{31}-1$)。

INT 类型

语法：INT

功能：与 INTEGER 相同。

9.4.2 日期时间数据类型

日期时间数据类型分为一般日期时间数据类型、时间间隔数据类型和时区数据类型三类，用于存储日期、时间和它们之间的间隔信息。

1. 一般日期时间数据类型

DATE 类型

```
语法：DATE
```

功能：DATE 类型包括年、月、日信息，定义了'-4712-01-01'和'9999-12-31'之间任何一个有效的格里高利日期。DM 支持儒略历，并考虑了历史上从儒略历转换至格里高利日期时的异常，不计算'1582-10-05'到'1582-10-14'之间的 10 天。

DM 支持 SQL92 标准或 SQL SERVER 的 DATE 字值，例如：DATE '1999-10-01'、'1999/10/01'或'1999.10.01'都是有效的 DATE 值，且彼此等价。年月日中第一个非 0 数值前的 0 亦可省略，例如'0001-01-01'等价于'1-1-1'。

TIME 类型

```
语法：TIME[(小数秒精度)]
```

功能：TIME 类型包括时、分、秒信息，定义了一个在'00:00:00.000000'和'23:59:59.999999'之间的有效时间。TIME 类型的小数秒精度规定了秒字段中小数点后面的位数，取值范围为 0～6，如果未定义，缺省精度为 0。

DM 支持 SQL92 标准或 SQL SERVER 的 TIME 字值，例如：TIME '09:10:21'，'09:10:21'或'9:10:21'都是有效的 TIME 值，且彼此等价。

9.5 DM_SQL 语言支持的表达式

DM 支持多种类型的表达式，包括数值表达式、字符串表达式、时间值表达式、时间间隔值表达式等。本节中引用的数据库实例请参见 DM 数据库示例库。

9.5.1 字符串表达式

连接 ||

```
语法：STR1 || STR2
```

(STR1 代表字符串 1，STR2 代表字符串 2)

连接操作符对两个运算数进行运算，其中每一个都是对属于同一字符集的字符串的求值。它以给定的顺序将字符串连接在一起，并返回一个字符串。其长度等于两个运算数长度之和。如果两个运算数中有一个是 NULL，则 NULL 等价为空串。

例

```
select '武汉' || ADDRESS1 from PERSON.ADDRESS WHERE ADDRESSID=3;
```

结果是：

武汉青山区青翠苑 1 号

9.5.2 时间值表达式

时间值表达式的结果为时间值类型，包括日期(DATE)类型，时间(TIME)类型和时间戳(TIMESTAMP)间隔类型。DM SQL 不是对于任何的日期时间和间隔运算数的组合都可以计算。如果任何一个运算数是 NULL，运算结果也是 NULL。下面列出了有效的可能性和结果的数据类型。

1. 日期+数值，日期-数值和数值+日期，得到日期

达梦数据库信创大赛资料

日期与数值的运算，等价于日期与一个 INTERVAL '数值' DAY 的时间间隔的运算。

例

```
SELECT CURDATE();
```

结果是：

2011-09-29 /* 假设该查询操作发生在 2011 年 9 月 29 日 */

例

```
SELECT CURDATE() + 2;
```

结果是：

2011-10-01

例

```
SELECT CURDATE() - 100;
```

结果是：

2011-06-21

2． 时间戳记+数值，时间戳记-数值和数值+时间戳记，得到时间戳记

时间戳记与数值的运算，将数值看作以天为单位，转化为一个 INTERVAL DAY TO SECOND(6)的时间间隔，然后进行运算。

例 时间戳加上 2.358 天。

```
SELECT TIMESTAMP '2003-09-29 08:59:59.123' + 2.358;
```

结果是：

2003-10-01 17:35:30.323000

第 10 章 数据查询语句

数据查询是数据库的核心操作，DM_SQL 语言提供了功能丰富的查询方式，满足实际应用需求。几乎所有的数据库操作均涉及到查询，因此熟练掌握查询语句的使用是数据库从业人员必须掌握的技能。

在 DM_SQL 语言中，有的定义语法中也包含查询语句，如视图定义语句、游标定义语句等。为了区别，我们将这类出现在其它定义语句中的查询语句称查询说明。

每种查询都有适用的场景，使用得当会大大提高查询效率。为方便用户的使用，本章对 DM_SQL 语言支持的查询方式进行讲解，测例中所用基表及各基表中预先装入的数据参见第 2 章，各例的建表者均为用户 SYSDBA。

查询语句的语法如下：

```
<查询表达式>::=
    <simple_select>|
    <select_clause> <ORDER BY 子句> |
<simple_select> ::=
    <query_exp_with>|
    <select_clause>  <UNION| EXCEPT | MINUS | INTERSECT > [ALL | DISTINCT | UNIQUE] <select_clause>
<select_clause>::=
    <simple_select>|
    (<查询表达式>)|
    (<select_clause>)
```

```
<ORDER BY 子句> ::= ORDER BY <order_by_list> [ASC | DESC] [NULLS FIRST|LAST]{,< order_by_list> [ASC | DESC] [NULLS FIRST|LAST]}
<order_by_list>::=
    <无符号整数> |
    <列说明> |
    <值表达式>
<query_exp_with>::=[<WITH 子句>] SELECT [ALL | DISTINCT | UNIQUE] <选择列表> <select_tail>
<选择列表> ::= [[<模式名>.]<基表名> | <视图名> .] * | <值表达式> [[AS] <列别名>]
        {,[[<模式名>.]<基表名> | <视图名>.] * | <值表达式> [[AS] <列别名>]}
<select_tail>::=
 <FROM 子句>
 [<WHERE 子句>]
 [<GROUP BY 子句>]
 [<HAVING 子句>]
<FROM 子句>::= FROM <表引用>{,<表引用>}
<表引用>::=<普通表>|<连接表>
<普通表>::=<普通表 1>|<普通表 2>|<普通表 3>
<普通表 1>::=<对象名> [[AS <别名>]]
<普通表 2>::=(<查询表达式>)[[AS <别名>]] [[AS] <表别名> [<新生列>]]
<普通表 3>::=[<模式名>.]<<基表名>|<视图名>>(<选择列>)[[AS <别名>]] [[AS <别名>]] [[AS] <表别名> [<派生列表>]]
<对象名>::=<本地对象> | <索引>
<本地对象>::=[<模式名>.]<基表名|视图名>
<索引>::=[<模式名>.]<基表名> INDEX <索引名>
<选择列>::=<列名>[{,<列名> }]
<派生列表>::=(<列名>[{,<列名>}])
 <连接表>::=[(]<限定连接>[)]
<限定连接>::=<表引用> [NATURAL] [<连接类型>] JOIN <<普通表>|(<连接表>)>
<连接类型>::=
    [<内外连接类型>]<INNER>|
    <内外连接类型>[<OUTER>]
<内外连接类型>::=LEFT|RIGHT|FULL
<连接条件>::=<条件匹配>|<列匹配>
<条件匹配>::=ON<搜索条件>
<列匹配>::=USING(<连接列列名>{, <连接列列名>})
<WHERE 子句> :: = WHERE <搜索条件>
<搜索条件>::=<逻辑表达式>
<GROUP BY 子句> ::= GROUP BY <group_by 项>{,<group_by 项>}
<group_by 项>::=<分组项>
<分组项>::=<值表达式>
<HAVING 子句> ::= HAVING <搜索条件>
```

参数

1. ALL　　　　返回所有被选择的行，包括所有重复的拷贝，缺省值为 ALL；

2. DISTINCT　从被选择出的具有重复行的每一组中仅返回一个这些行的拷贝，与 UNIQUE 等价。对于集合算符：UNION，缺省值为 DISTINCT，DISTINCT 与 UNIQUE 等价；对于 EXCEPT/MINUS 和 INTERSECT：操作的两个表中数据类型和个数要完全一致。其中，EXCEPT 和 MINUS 集合算符功能完全一样，返回两个集合的差集；INTERSECT 返回两个集合的交集（去除重复记录）；

3. <模式名>　　被选择的表和视图所属的模式，缺省为当前模式；

4. <基表名>　　被选择数据的基表的名称；

5. <视图名>　　被选择数据的视图的名称；

6. *　　指定对象的所有的列；

7. <值表达式>　可以为一个<列引用>、<集函数>、<函数>、<标量子查询>或<计算表达式>等等；

8. <列别名>　　为列表达式提供不同的名称，使之成为列的标题，列别名不会影响实际的名称，别名在该查询中被引用；

9. <相关名>　　给表、视图提供不同的名字，经常用于求子查询和相关查询的目的；

10. <列名>　　指明列的名称；

11. <WHERE 子句>　限制被查询的行必须满足条件，如果忽略该子句，DM 从在 FROM 子句中的表、视图中选取所有的行；其中，<WHERE CURRENT OF 子句>专门用于游标更新、删除中，用来限定更新、删除与游标有关的数据行。

12. <HAVING 子句>　限制所选择的行组所必须满足的条件，缺省为恒真，即对所有的组都满足该条件；

13. <无符号整数>　　指明了要排序的<值表达式>在 SELECT 后的序列号；

14. <列说明>　　排序列的名称；

15. ASC　　指明为升序排列，缺省为升序；

16. DESC　　指明为降序排列；

17. NULLS FIRST　指定排序列的 NULL 值放在最前面，不受 ASC 和 DESC 的影响，缺省的是 NULLS FIRST；

18. NULLS LAST　指定排序列的 NULL 值放在最后面，不受 ASC 和 DESC 的影响；

使用说明

1. <选择列表>中最多可包含 1024 个查询项，且查询记录的长度限制不能超过块长的一半；

2. <FROM 子句>中最多可引用 100 张表；

3. WHERE<搜索条件> 用于设置对于行的检索条件。不在规定范围内的任何行都从结果集中去除；

4. 查询语句调用的函数中，不能包含任何增删改操作（包括函数间接调用其它过程\函数产生的增删改操作）；

5. EXCEPT/MINUS/INTERSECT 集合运算中，查询列不能含有 BLOB、CLOB 或 IMAGE、TEXT 等大字段类型；

10.1 单表查询

SELECT 语句仅从一个表/视图中检索数据，称单表查询。即<FROM 子句>中<普通表>使用的是[<模式名>.]<基表名|视图名>。

达梦数据库信创大赛资料

10.1.1 简单查询

例 查询所有图书的名字、作者及当前销售价格，并消去重复。

```
SELECT  DISTINCT  NAME, AUTHOR, NOWPRICE  FROM  PRODUCTION .PRODUCT;
```

其中，DISTINCT 保证重复的行将从结果中去除。若允许有重复的元组，改用 ALL 来替换 DISTINCT，或直接去掉 DISTINCT 即可。

查询结果如表 4.1.1 所示。(注：除带 Order By 的查询外，本书所示查询结果中各元组的顺序与实际输出结果中的元组顺序不一定一致。)

表 4.1.1

NAME	AUTHOR	NOWPRICE
长征	王树增	37.7000
工作中无小事	陈满麒	11.4000
红楼梦	曹雪芹，高鹗	15.2000
老人与海	海明威	6.1000
鲁迅文集(小说、散文、杂文)全两册	鲁迅	20.0000
射雕英雄传(全四册)	金庸	21.7000
数据结构(C 语言版)(附光盘)	严蔚敏，吴伟民	25.5000
水浒传	施耐庵，罗贯中	14.3000
突破英文基础词汇	刘毅	11.1000
噼里啪啦丛书(全 7 册)	(日)佐佐木洋子	42.0000

当用户需要查出所有列的数据，且各列的显示顺序与基表中列的顺序也完全相同时，为了方便用户提高工作效率，SQL 语言允许用户将 SELECT 后的<值表达式>省略为*。

```
SELECT  *  FROM  PERSON.PERSON;
```

等价于：

```
SELECT  PERSONID, NAME, SEX, EMAIL, PHONE  FROM  PERSON.PERSON;
```

其查询结果是模式 PERSON 中基表 PERSON 的一份拷贝，结果从略。

例 1）查询 tt 表中有的，kk 表中没有的数据；2）查询 tt 表和 kk 表都有的数据。

```
CREATE TABLE TT(A INT);
INSERT INTO TT VALUES(5);
INSERT INTO TT VALUES(6);
INSERT INTO TT VALUES(7);

CREATE TABLE KK(A INT);
INSERT INTO KK VALUES(5);
INSERT INTO KK VALUES(5);
INSERT INTO KK VALUES(6);
INSERT INTO KK VALUES(8);
```

1) 使用 MINUS 或 EXCEPT 查询 tt 表中有的，kk 表中没有的数据。

```
SELECT * FROM TT MINUS SELECT * FROM KK;  等价于 SELECT * FROM TT EXCEPT SELECT * FROM KK;
```

其查询结果是：

```
A
```

```
7
```

2）使用 INTERSECT 查询 TT 表中和 KK 表中都有的数据。

```
SELECT * FROM TT INTERSECT SELECT * FROM KK;
```

其查询结果是：

```
A
5
6
```

10.1.2 带条件查询

带条件查询是指在指定表中查询出满足条件的元组。该功能是在查询语句中使用 WHERE 子句实现的。WHERE 子句常用的查询条件由谓词和逻辑运算符组成。谓词指明了一个条件，该条件求解后，结果为一个布尔值：真、假或未知。

逻辑算符有：AND，OR，NOT。

谓词包括比较谓词(＝、＞、＜、＞＝、＜＝、＜＞)，BETWEEN 谓词、IN 谓词、LIKE 谓词、NULL 谓词、EXISTS 谓词。

1．使用比较谓词的查询

当使用比较谓词时，数值数据根据它们代数值的大小进行比较，字符串的比较则按序对同一顺序位置的字符逐一进行比较。若两字符串长度不同，短的一方应在其后增加空格，使两串长度相同后再作比较。

例 给出当前销售价格在 10～20 元之间的所有图书的名字、作者、出版社和当前价格。

```
SELECT  NAME, AUTHOR, PUBLISHER, NOWPRICE  FROM  PRODUCTION.PRODUCT
WHERE  NOWPRICE>=10  AND  NOWPRICE<=20;
```

查询结果如表 4.1.2 所示。

表 4.1.2

NAME	AUTHOR	PUBLISHER	NOWPRICE
红楼梦	曹雪芹，高鹗	中华书局	15.2000
水浒传	施耐庵，罗贯中	中华书局	14.3000
鲁迅文集(小说、散文、杂文)全两册	鲁迅		20.0000
工作中无小事	陈满麒	机械工业出版社	11.4000
突破英文基础词汇	刘毅	外语教学与研究出版社	11.1000

2．使用 BETWEEN 谓词的查询

例 给出当前销售价格在 10～20 元之间的所有图书的名字、作者、出版社和当前价格。

```
SELECT  NAME, AUTHOR, PUBLISHER, NOWPRICE  FROM  PRODUCTION.PRODUCT
WHERE  NOWPRICE  BETWEEN  10  AND  20;
```

此例查询与上例完全等价，查询结果如上表所示。在 BETWEEN 谓词前面可以使用 NOT，以表示否定。

3．使用 IN 谓词的查询

谓词 IN 可用来查询某列值属于指定集合的元组。

例 查询出版社为中华书局或人民文学出版社出版的图书名称与作者信息。

```
SELECT  NAME, AUTHOR  FROM  PRODUCTION.PRODUCT
WHERE  PUBLISHER  IN  ('中华书局', '人民文学出版社');
```

查询结果如表 4.1.3 所示。

表 4.1.3

NAME	AUTHOR
红楼梦	曹雪芹，　高鹗
水浒传	施耐庵，　罗贯中
长征	王树增

在 IN 谓词前面也可用 NOT 表示否定。

4. 使用 LIKE 谓词的查询

LIKE 谓词一般用来进行字符串的匹配。我们先用实例来说明 LIKE 谓词的使用方法。

例 查询第一通讯地址中第四个字开始为“关山”且以 202 结尾的地址。

```
SELECT  ADDRESSID,  ADDRESS1,  CITY,  POSTALCODE  FROM  PERSON.ADDRESS
WHERE  ADDRESS1  LIKE  '___关山%202';
```

查询结果如表 4.1.4 所示。

表 4.1.4

ADDRESSID	ADDRESS1	CITY	POSTALCODE
13	洪山区关山春晓 55-1-202	武汉市洪山区	430073
14	洪山区关山春晓 10-1-202	武汉市洪山区	430073
15	洪山区关山春晓 11-1-202	武汉市洪山区	430073

由上例可看出，LIKE 谓词的一般使用格式为：

```
<列名>  LIKE  <匹配字符串常数>
```

其中，<列名>必须是可以转化为字符类型的数据类型的列。对于一个给定的目标行，如果指定列值与由<匹配字符串常数>给出的内容一致，则谓词结果为真。<匹配字符串常数>中的字符可以是一个完整的字符串，也可以是百分号“%”和下划线“_”，“%”和“_”称通配符。“%”代表任意字符串(也可以是空串)；“_”代表任何一个字符。

因此，上例中的 SELECT 语句将从 ADDRESS 表中检索出第一通讯地址中第四个字开始为“关山”且以 202 结尾的地址情况。从该例我们可以看出 LIKE 谓词是非常有用的。使用 LIKE 谓词可以找到所需要的但又记不清楚的那样一些信息。这种查询称模糊查询或匹配查询。为了加深对 LIKE 谓词的理解，下面我们再举几例：

```
ADDRESS1  LIKE  '%洪山%'
```

如果 ADDRESS1 的值含有字符“洪山”，则该谓词取真值。

```
POSTALCODE  LIKE  '43__7_'
```

如果 POSTALCODE 的值由六个字符组成且前两个字符为 43，第五个字符为 7，则该谓词取真值。

```
CITY  LIKE  '%汉阳_'
```

如果 CITY 的值中倒数第三和第二个字为汉阳，则该谓词取真值。

```
ADDRESS1  NOT  LIKE  '洪山%'
```

如果 ADDRESS1 的值的前两个字不是洪山，则该谓词取真值。

阅读以上的例子，读者可能就在想这样一个问题：如果<匹配字符串常数>中所含“%”和“_”不是作通配符，而只是作一般字符使用应如何表达呢？为解决这一问题，SQL 语句对 LIKE 谓词专门提供了对通配符“%”和“_”的转义说明，这时 LIKE 谓语使用格式为：

```
<列名>  LIKE  '<匹配字符串常数>'  [ESCAPE  <转义字符>]
```

其中，<转义字符>指定了一个字符，当该字符出现在<匹配字符串常数>中时，用以指明紧跟其后的“%”或“_”不是通配符而仅作一般字符使用。

例 查询第一通讯地址以 C1_501 结尾的地址，则 LIKE 谓词应为：

```
SELECT  ADDRESSID,  ADDRESS1,  CITY,  POSTALCODE  FROM  PERSON.ADDRESS
WHERE  ADDRESS1  LIKE  '%C1*_501'  ESCAPE  '*';
```

在此例中，*被定义为转义字符，因而在<匹配字符串常数>中*号后的下划线不再作通配符，而是普通字符。

查询结果如表 4.1.5 所示。

表 4.1.5

ADDRESSID	ADDRESS1	CITY	POSTALCODE
16	洪山区光谷软件园 C1_501	武汉市洪山区	430073

为避免错误，转义字符一般不要选通配符“%”、“_”或在<匹配字符串常数>中已出现的字符。

6. 使用 NULL 谓词的查询

空值是未知的值。当列的类型为数值类型时，NULL 并不表示 0；当列的类型为字符串类型时，NULL 也并不表示空串。因为 0 和空串也是确定值。NULL 只能是一种标识，表示它在当前行中的相应列值还未确定或未知，对它的查询也就不能使用比较谓词而须使用 NULL 谓词。

例 查询哪些人员的 EMAIL 地址为 NULL。

```
SELECT  NAME,  SEX,  PHONE  FROM  PERSON.PERSON
WHERE  EMAIL  IS  NULL;
```

在 NULL 谓词前，可加 NOT 表示否定。

7. 组合逻辑

可以用逻辑算符(AND，OR，NOT)与各种谓词相组合生成较复杂的条件查询。

例 查询当前销售价格低于 15 元且折扣低于 7 或出版社为人民文学出版社的图书名称和作者。

```
SELECT  NAME,  AUTHOR  FROM  PRODUCTION.PRODUCT
WHERE  NOWPRICE  <  15  AND  DISCOUNT  <  7  OR  PUBLISHER='人民文学出版社';
```

查询结果如表 4.1.6 所示。

表 4.1.6

NAME	AUTHOR
老人与海	海明威
长征	王树增
工作中无小事	陈满麒

10.1.3 集函数

为了进一步方便用户的使用，增强查询能力，SQL 语言提供了多种内部集函数。集函数又称库函数，当根据某一限制条件从表中导出一组行集时，使用集函数可对该行集作统计操作。

1) 函数分类

集函数可分为 10 类：

1. COUNT(*)；
2. 相异集函数 AVG|MAX|MIN|SUM|COUNT(DISTINCT<列名>)；
3. 完全集函数 AVG|MAX|MIN|COUNT|SUM([ALL]<值表达式>)；

2) 使用说明

在使用集函数时要注意以下几点：

1. 相异集函数与完全集函数的区别是：相异集函数是对表中的列值消去重复后再作集函数运算，而完全集函数是对包含列名的值表达式作集函数运算且不消去重复。缺省情况下，集函数均为完全集函数；

2. 集函数中的自变量可以是集函数，但最多只能嵌套 2 层。嵌套分组函数的时候，需要使用 GROUP BY；

3. **AVG**、**SUM** 的参数必须为数值类型；**MAX**、**MIN** 的结果数据类型与参数类型保持一致；对于 **SUM** 函数，如果参数类型为 BYTE、BIT、SMALLINT 或 INTEGER，那么结果类型为 INTEGER，如果参数类型为 NUMERIC、DECIMAL、FLOAT 和 DOUBLE PRECISION，那么结果类型为 DOUBLE PRECISION；COUNT 结果类型统一为 BIGINT；

对于 **AVG** 函数，其参数类型与结果类型对应关系如表 4.1.7 所示：

表 4.1.7

参数类型	结果类型
tinyint	dec(38,6)
smallint	dec(38,6)
int	dec(38,6)
bigint	dec(38,6)
float	double
double	double
dec(x,y)	number

达梦数据库信创大赛资料

3)　　举例说明

下面按集函数的功能分别举例说明。

1.　求最大值集函数 MAX 和求最小值集函数 MIN

例 查询折扣小于 7 的图书中现价最低的价格。

```
SELECT  MIN(NOWPRICE)  FROM  PRODUCTION.PRODUCT
WHERE  DISCOUNT < 7;
```

查询结果为：6.1000

需要说明的是：SELECT 后使用集函数 MAX 和 MIN 得到的是一个最大值和最小值，因而 SELECT 后不能再有列名出现，如果有只能出现在集函数中。如：

```
SELECT  NAME,MIN(NOWPRICE)  FROM  PRODUCTION.PRODUCT;
```

DM 系统会报错，因为 NAME 是一个行集合，而最低价格是唯一值。

至于 MAX 的使用格式与 MIN 是完全一样的，读者可以自己举一反三。

2.　求平均值集函数 AVG 和总和集函数 SUM

例 求折扣小于 7 的图书的平均现价。

```
SELECT  AVG(NOWPRICE)  FROM PRODUCTION.PRODUCT WHERE  DISCOUNT < 7;
```

查询结果为：23.15

例 求折扣大于 8 的图书的总价格。

```
SELECT  SUM(NOWPRICE) FROM  PRODUCTION.PRODUCT WHERE   DISCOUNT >8;
```

查询结果为：25.5

3.　求总个数集函数 COUNT

例 查询已登记供应商的个数。

```
SELECT  COUNT(*)  FROM  PURCHASING.VENDOR;
```

查询结果为：12

由此例可看出，COUNT(*)的结果是 VENDOR 表中的总行数，由于主关键字不允许有相同值，因此，它不需要使用保留字 DISTINCT。

例 查询目前销售的图书的出版商的个数。

```
SELECT  COUNT(DISTINCT  PUBLISHER)  FROM  PRODUCTION.PRODUCT;
```

查询结果为：9

由于一个出版商可出版多种图书，因而此例中一定要用 DISTINCT 才能得到正确结果。

10.2 连接查询

如果一个查询包含多个表(>=2)，则称这种方式的查询为连接查询。即<FROM 子句>中使用的是<连接表>。DM 的连接查询方式包括：交叉连接(cross join)、自然连接(natural join）、内连接（inner）、外连接（outer）。下面分别举例说明。

10.2.1 JOIN … ON

这是一种连接查询的常用写法，说明是一个连接查询。JOIN 关键字指定连接的两张表，ON 子句指定连接条件表达式，其中不允许出现 ROWNUM。具体采用何种连接方式，由数据库内部分析确定。

例 查询销售人员的入职时间和去年销售总额。

```
SELECT  T1.HAIRDATE,
T2.SALESLASTYEAR
FROM  RESOURCES.EMPLOYEE  T1 JOIN  SALES.SALESPERSON  T2
ON  T1.EMPLOYEEID=T2.EMPLOYEEID;
```

查询结果如表 4.2.5 所示。

表 4.2.5

HAIRDATE	SALESLASTYEAR
2002-5-2	10.0000
2002-5-2	20.0000

10.2.2 外连接(OUTER JOIN)

外连接对结果集进行了扩展，会返回一张表的所有记录，对于另一张表无法匹配的字段用 NULL 填充返回。DM 数据库支持三种方式的外连接：左外连接、右外连接、全外连接。

外连接中常用到的术语：左表、右表。根据表所在外连接中的位置来确定，位于左侧的表，称为左表；位于右侧的表，称为右表。例如 SELECT * FROM T1 LEFT JOIN T2 ON T1.C1=T2.D1，T1 表为左表，T2 表为右表。

返回所有记录的表根据外连接的方式而定。

1. 左外连接：返回左表所有记录；
2. 右外连接：返回右表所有记录；
3. 全外连接：返回两张表所有记录。处理过程为分别对两张表进行左外连接和右外连接，然后合并结果集。

在左外连接和右外连接中，如果需要对未能匹配的缺失数据进行填充，可以使用分区外连接(PARTITION OUTER JOIN)，分区外连接通常用于处理稀疏数据以得到分析报表。

下面举例说明。

例 从 PRODUCT_CATEGORY、PRODUCT_SUBCATEGORY 中查询图书的所有目录名称和子目录名称，包括没有子目录的目录。

```
SELECT T1.NAME, T2.NAME
FROM  PRODUCTION.PRODUCT_CATEGORY T1  LEFT OUTER JOIN
PRODUCTION.PRODUCT_SUBCATEGORY T2
ON T1.PRODUCT_CATEGORYID = T2.PRODUCT_CATEGORYID;
```

查询结果如表 4.2.8 所示。

表 4.2.8

NAME	NAME
小说	世界名著
小说	武侠

小说	科幻
小说	四大名著
小说	军事
小说	社会
文学	文集
文学	纪实文学
文学	文学理论
文学	中国古诗词
文学	中国现当代诗
文学	戏剧
文学	民间文学
计算机	计算机理论
计算机	计算机体系结构
计算机	操作系统
计算机	程序设计
计算机	数据库
计算机	软件工程
计算机	信息安全
计算机	多媒体
英语	英语词汇
英语	英语语法
英语	英语听力
英语	英语口语
英语	英语阅读
英语	英语写作
管理	行政管理
管理	项目管理
管理	质量管理与控制
管理	商业道德
管理	经营管理
管理	财务管理
少儿	幼儿启蒙
少儿	益智游戏
少儿	童话
少儿	卡通
少儿	励志
少儿	少儿英语
金融	NULL

例 从 PRODUCT_CATEGORY、PRODUCT_SUBCATEGORY 中查询图书的目录名称和所有子目录名称，包括没有目录的子目录。

```
SELECT T1.NAME, T2.NAME
FROM  PRODUCTION.PRODUCT_CATEGORY T1  RIGHT OUTER JOIN
PRODUCTION.PRODUCT_SUBCATEGORY T2
```

达梦数据库优创大赛资料

```
ON T1.PRODUCT_CATEGORYID = T2.PRODUCT_CATEGORYID;
```

查询结果如表 4.2.9 所示。

表 4.2.9

NAME	NAME
小说	世界名著
小说	武侠
小说	科幻
小说	四大名著
小说	军事
小说	社会
文学	文集
文学	纪实文学
文学	文学理论
文学	中国古诗词
文学	中国现当代诗
文学	戏剧
文学	民间文学
计算机	计算机理论
计算机	计算机体系结构
计算机	操作系统
计算机	程序设计
计算机	数据库
计算机	软件工程
计算机	信息安全
计算机	多媒体
英语	英语词汇
英语	英语语法
英语	英语听力
英语	英语口语
英语	英语阅读
英语	英语写作
管理	行政管理
管理	项目管理
管理	质量管理与控制
管理	商业道德
管理	经营管理
管理	财务管理
少儿	幼儿启蒙
少儿	益智游戏
少儿	童话
少儿	卡通
少儿	励志
少儿	少儿英语

达梦数据库信创大赛资料

NULL	历史

例 从 PRODUCT_CATEGORY、PRODUCT_SUBCATEGORY 中查询图书的所有目录名称和所有子目录名称。

```
SELECT T1.NAME, T2.NAME
FROM  PRODUCTION.PRODUCT_CATEGORY T1  FULL OUTER JOIN
PRODUCTION.PRODUCT_SUBCATEGORY T2
ON T1.PRODUCT_CATEGORYID = T2.PRODUCT_CATEGORYID;
```

查询结果如表 4.2.10 所示。

表 4.2.10

NAME	NAME
小说	世界名著
小说	武侠
小说	科幻
小说	四大名著
小说	军事
小说	社会
文学	文集
文学	纪实文学
文学	文学理论
文学	中国古诗词
文学	中国现当代诗
文学	戏剧
文学	民间文学
计算机	计算机理论
计算机	计算机体系结构
计算机	操作系统
计算机	程序设计
计算机	数据库
计算机	软件工程
计算机	信息安全
计算机	多媒体
英语	英语词汇
英语	英语语法
英语	英语听力
英语	英语口语
英语	英语阅读
英语	英语写作
管理	行政管理
管理	项目管理
管理	质量管理与控制
管理	商业道德
管理	经营管理
管理	财务管理

达梦数据库信创大赛资料

少儿	幼儿启蒙
少儿	益智游戏
少儿	童话
少儿	卡通
少儿	励志
少儿	少儿英语
金融	NULL
NULL	历史

外连接还有一种写法，在连接条件或 WHERE 条件中，在列后面增加(+)指示左外连接或者右外连接。如果表 A 和表 B 连接，连接条件或者 where 条件中，A 的列带有(+)后缀，则认为是 B LEFT JOIN A。如果用户的(+)指示引起了外连接环，则报错。下面举例说明。

例 从 PRODUCT_CATEGORY、PRODUCT_SUBCATEGORY 中查询图书的目录名称和所有子目录名称，包括没有目录的子目录。

```
SELECT T1.NAME, T2.NAME
FROM PRODUCTION.PRODUCT_CATEGORY T1, PRODUCTION.PRODUCT_SUBCATEGORY T2
WHERE T1.PRODUCT_CATEGORYID(+) = T2.PRODUCT_CATEGORYID;
```

查询结果如上表 4.2.9 所示。

例 新建产品区域销售统计表 SALES.SALESREGION 并插入数据。

```
CREATE TABLE SALES.SALESREGION(REGION CHAR(10), PRODUCTID INT, AMOUNT INT);
INSERT INTO SALES.SALESREGION VALUES('大陆', 2, 19800);
INSERT INTO SALES.SALESREGION VALUES('大陆', 4, 20090);
INSERT INTO SALES.SALESREGION VALUES('港澳台', 6, 5698);
INSERT INTO SALES.SALESREGION VALUES('外国', 9, 3756);
COMMIT;
```

统计每个产品在各个区域的销售量，没有销售则显示 NULL，此时可使用 PARTITON OUTER JOIN 将稀疏数据转为稠密数据。

```
SELECT A.PRODUCTID, B.REGION, B.AMOUNT
FROM PRODUCTION.PRODUCT A LEFT JOIN SALES.SALESREGION B
PARTITION BY(B.REGION) ON A.PRODUCTID=B.PRODUCTID
ORDER BY A.PRODUCTID, B.REGION;
```

查询结果如表 4.2.10 所示。

表 4.2.10

PRODUCTID	REGION	AMOUNT
1	大陆	NULL
1	港澳台	NULL
1	外国	NULL
2	大陆	19800
2	港澳台	NULL
2	外国	NULL
3	大陆	NULL
3	港澳台	NULL
3	外国	NULL
4	大陆	20090
4	港澳台	NULL

4	外国	NULL
5	大陆	NULL
5	港澳台	NULL
5	外国	NULL
6	大陆	NULL
6	港澳台	5698
6	外国	NULL
7	大陆	NULL
7	港澳台	NULL
7	外国	NULL
8	大陆	NULL
8	港澳台	NULL
8	外国	NULL
9	大陆	NULL
9	港澳台	NULL
9	外国	3756
10	大陆	NULL
10	港澳台	NULL
10	外国	NULL

10.3 子查询

在 DM_SQL 语言中，一个 SELECT-FROM-WHERE 语句称为一个查询块，如果在一个查询块中嵌套一个或多个查询块，我们称这种查询为子查询。子查询会返回一个值(标量子查询)或一个表(表子查询)。它通常采用(SELECT…)的形式嵌套在表达式中。子查询语法如下：

```
<子查询> ::= (<查询表达式>)
```

即子查询是嵌入括弧的<查询表达式>，而这个<查询表达式>通常是一个 SELECT 语句。它有下列限制：

1． 在子查询中不得有 ORDER BY 子句；

2． 子查询允许 TEXT 类型与 CHAR 类型值比较。比较时，取出 TEXT 类型字段的最多 8188 字节与 CHAR 类型字段进行比较；如果比较的两字段都是 TEXT 类型，则最多取 300*1024 字节进行比较；

3． 子查询不能包含在集函数中；

4． 在子查询中允许嵌套子查询。

按子查询返回结果的形式，DM 子查询可分为两大类：

1. 标量子查询：只返回一行一列；
2. 表子查询：可返回多行多列。

10.3.1 表子查询

和标量子查询不同的是，表子查询的查询结果可以是多行多列。

一般情况下，表子查询类似标量子查询，单列构成了表子查询的选择清单，但它的查询结果允许返回多行。可以从上下文中区分出表子查询：在其前面始终有一个只对表子查询的算符：<比较算符>ALL、<比较算符>ANY(或是其同义词<比较算符> SOME)、IN 和 EXISTS。

其中，在 IN/NOT IN 表子查询的情况下，DM 支持查询结果返回多列。

例 查询职务为销售代表的员工的编号、今年销售总额和去年销售总额。

```
SELECT  EMPLOYEEID, SALESTHISYEAR, SALESLASTYEAR
FROM   SALES.SALESPERSON
WHERE  EMPLOYEEID IN
( SELECT  EMPLOYEEID
  FROM  RESOURCES.EMPLOYEE
  WHERE  TITLE = '销售代表'
);
```

查询结果如表 4.3.1 所示。

表 4.3.1

EMPLOYEEID	SALESTHISYEAR	SALESLASTYEAR
4	8.0000	10.0000
5	8.0000	20.0000

该查询语句的求解方式是：首先通过子查询"SELECT EMPLOYEEID FROM RESOURCES.EMPLOYEE WHERE TITLE = '销售代表'"查到职务为销售代表的 EMPLOYEEID 的集合，然后，在 SALESPERSON 表中找到与子查询结果集中的 EMPLOYEEID 所对应员工的 SALESTHISYEAR 和 SALESLASTYEAR。

在带有子查询的查询语句中，通常也将子查询称内层查询或下层查询。由于子查询还可以嵌套子查询，相对于下一层的子查询，上层查询又称为父查询或外层查询。

由于 DM_SQL 语言所支持的嵌套查询功能可以将一系列简单查询构造成复杂的查询，从而有效地增强了 DM_SQL 语句的查询功能。以嵌套的方式构造语句是 DM_SQL 的"结构化"的特点。

需要说明的是：上例的外层查询只能用 IN 谓词而不能用比较算符"="，因为子查询的结果包含多个元组，除非能确定子查询的结果只有一个元组时，才可用等号比较。上例语句也可以用连接查询的方式实现。

```
SELECT T1.EMPLOYEEID, T1.SALESTHISYEAR, T1.SALESLASTYEAR
FROM  SALES.SALESPERSON T1 , RESOURCES.EMPLOYEE T2
WHERE T1.EMPLOYEEID = T2.EMPLOYEEID AND T2.TITLE = '销售代表';
```

例 查询对目录名为小说的图书进行评论的人员名称和评论日期。

采用子查询嵌套方式写出以下查询语句：

```
SELECT  DISTINCT NAME, REVIEWDATE
FROM   PRODUCTION.PRODUCT_REVIEW
WHERE  PRODUCTID  IN
( SELECT  PRODUCTID
  FROM  PRODUCTION.PRODUCT
  WHERE  PRODUCT_SUBCATEGORYID IN
  ( SELECT  PRODUCT_SUBCATEGORYID
    FROM   PRODUCTION.PRODUCT_SUBCATEGORY
    WHERE  PRODUCT_CATEGORYID IN
    ( SELECT  PRODUCT_CATEGORYID
```

```
          FROM  PRODUCTION.PRODUCT_CATEGORY
          WHERE  NAME = '小说'
        )
      )
    );
```

查询结果如表 4.3.2 所示。

表 4.3.2

NAME	REVIEWDATE
刘青	2007-05-06
桑泽恩	2007-05-06

该语句采用了四层嵌套查询方式，首先通过最内层子查询从 PRODUCT_CATEGORY 中查出目录名为小说的目录编号，然后从 PRODUCT_SUBCATEGORY 中查出这些目录编号对应的子目录编号，接着从 PRODUCT 表中查出这些子目录编号对应的图书的编号，最后由最外层查询查出这些图书编号对应的评论人员和评论日期。

此例也可用四个表的连接来完成。

从上例可以看出，当查询涉及到多个基表时，嵌套子查询与连接查询相比，前者由于是逐步求解，层次清晰，易于阅读和理解，具有结构化程序设计的优点。

在许多情况下，外层子查询与内层子查询常常引用同一个表，如下例所示。

例 查询当前价格低于红楼梦的图书的名称、作者和当前价格。

```
SELECT  NAME, AUTHOR,  NOWPRICE
FROM  PRODUCTION.PRODUCT
WHERE  NOWPRICE < ( SELECT NOWPRICE FROM PRODUCTION.PRODUCT
                    WHERE NAME = '红楼梦');
```

查询结果如表 4.3.3 所示。

表 4.3.3

NAME	AUTHOR	NOWPRICE
水浒传	施耐庵，罗贯中	14.3000
老人与海	海明威	6.1000
工作中无小事	陈满麒	11.4000
突破英文基础词汇	刘毅	11.1000

此例的子查询与外层查询尽管使用了同一表名，但作用是不一样的。子查询是在该表中红楼梦的图书价格，而外查询是在 PRODUCT 表 NOWPRICE 列查找小于该值的集合，从而得到这些值所对应的名称和作者。DM_SQL 语言允许为这样的表引用定义别名：

```
SELECT  NAME, AUTHOR,  NOWPRICE
FROM  PRODUCTION.PRODUCT  T1
WHERE  T1.NOWPRICE < ( SELECT  T2.NOWPRICE
FROM  PRODUCTION.PRODUCT  T2
                       WHERE  T2.NAME = '红楼梦');
```

该语句也可以采用连接方式实现：

```
SELECT  T1.NAME, T1.AUTHOR, T1.NOWPRICE
FROM  PRODUCTION.PRODUCT T1 ,  PRODUCTION.PRODUCT T2
WHERE  T2.NAME = '红楼梦' AND T1.NOWPRICE < T2.NOWPRICE;
```

例 查询图书的出版社和产品供应商名称相同的图书编号和名称。

```
SELECT  T1.PRODUCTID, T1.NAME
FROM  PRODUCTION.PRODUCT T1, PRODUCTION.PRODUCT_VENDOR T2
WHERE  T1.PRODUCTID = T2.PRODUCTID  AND T1.PUBLISHER = ANY
( SELECT  NAME  FROM  PURCHASING.VENDOR T3
  WHERE  T2.VENDORID = T3.VENDORID);
```

其结果如表 4.3.4 所示。

表 4.3.4

PRODUCTID	NAME
1	红楼梦
2	水浒传
3	老人与海
4	射雕英雄传(全四册)
7	数据结构(C 语言版)(附光盘)
8	工作中无小事
9	突破英文基础词汇
10	噼里啪啦丛书(全 7 册)

此例有一点需要注意：子查询的 WHERE 子句涉及到 PRODUCT_VENDOR.VENDORID(即 T2.VENDORID)，但是其 FROM 子句中却没有提到 PRODUCT_VENDOR。在外部子查询 FROM 子句中命名了 PRODUCT_VENDOR——这就是外部引用。当一个子查询含有一个外部引用时，它就与外部语句相关联，称这种子查询为相关子查询。

例 查询图书的出版社和产品供应商名称不相同的图书编号和名称。

```
SELECT  T1.PRODUCTID, T1.NAME
FROM  PRODUCTION.PRODUCT T1
WHERE  T1.PUBLISHER <> ALL(SELECT  NAME  FROM  PURCHASING.VENDOR );
```

其结果如表 4.3.5 所示。

表 4.3.5

PRODUCTID	NAME
5	鲁迅文集(小说、散文、杂文)全两册
6	长征

10.4 合并查询结果

DM 提供了一种集合运算符：UNION，这种运算符将两个或多个查询块的结果集合并为一个结果集输出。语法如下：

语法格式

```
<查询表达式>
UNION [ALL][DISTINCT]
[ ( ]<查询表达式> [ ) ];
```

使用说明

1. 每个查询块的查询列数目必须相同；
2. 每个查询块对应的查询列的数据类型必须兼容；
3. 在 UNION 后的可选项关键字 ALL 的意思是保持所有重复，而没有 ALL 的情况下

表示删除所有重复；

4. 在 UNION 后的可选项关键字 DISTINCT 的意思是删除所有重复。缺省值为 DISTINCT。

举例说明

例 查询所有图书的出版商，查询所有图书供应商的名称，将两者连接，并去掉重复行。

```
SELECT PUBLISHER FROM PRODUCTION.PRODUCT
UNION
SELECT NAME FROM PURCHASING.VENDOR ORDER BY 1;
```

查询结果如表 4.5.1 所示。

表 4.5.1

PUBLISHER
21 世纪出版社
北京十月文艺出版社
长江文艺出版社
广州出版社
机械工业出版社
清华大学出版社 （注：末尾含空格）
清华大学出版社（注：末尾不含空格）
人民文学出版社
人民邮电出版社
上海出版社
上海画报出版社
外语教学与研究出版社
文学出版社
中华书局

例 UNION ALL。

```
SELECT  PUBLISHER  FROM  PRODUCTION.PRODUCT
UNION ALL
SELECT  NAME  FROM  PURCHASING.VENDOR ORDER BY 1;
```

查询结果如表 4.5.2 所示。

表 4.5.2

PUBLISHER
21 世纪出版社
21 世纪出版社
北京十月文艺出版社
长江文艺出版社
广州出版社
广州出版社
机械工业出版社
机械工业出版社

达梦数据库信创大赛资料

清华大学出版社
清华大学出版社
人民文学出版社
人民邮电出版社
上海出版社
上海出版社
上海画报出版社
外语教学与研究出版社
外语教学与研究出版社
文学出版社
中华书局
中华书局
中华书局

10.5 分组查询

GROUP BY 子句逻辑地将由 WHERE 子句返回的临时结果重新编组。结果是行的集合，一组内一个分组列的所有值都是相同的。HAVING 子句用于为组设置检索条件。

10.5.1 GROUP BY 子句的使用

GROUP BY 子句是 SELECT 语句的可选项部分。它定义了分组表。GROUP BY 子句语法如下：

```
<GROUP BY 子句> ::= GROUP BY <group_by 项>{,<group_by 项>}
<group_by 项>::=<分组项>
<分组项>::= <值表达式>
```

GROUP BY 定义了分组表：行组的集合，其中每一个组由其中所有分组列的值都相等的行构成。

例 统计每个部门的员工数。

```
SELECT DEPARTMENTID,COUNT(*) FROM RESOURCES.EMPLOYEE_DEPARTMENT GROUP BY DEPARTMENTID;
```

查询结果如表 4.6.1 所示。

表 4.6.1

DEPARTMENTID	COUNT(*)
4	1
3	1
2	3
1	2

系统执行此语句时，首先将 EMPLOYEE_DEPARTMENT 表按 DEPARTMENTID 列进行分组，相同的 DEPARTMENTID 为一组，然后对每一组使用集函数 COUNT(*)，统计该组内的记录个数，如此继续，直到处理完最后一组，返回查询结果。

如果存在 WHERE 子句，系统先根据 WHERE 条件进行过滤，然后对满足条件的记录进行

分组。

此外，GROUP BY 不会对结果集排序。如果需要排序，可以使用 ORDER BY 子句。

例 求小说类别包含的子类别所对应的产品数量，并按子类别编号的升序排列。

```
SELECT A1.PRODUCT_SUBCATEGORYID AS 子分类编号,A3.NAME AS 子分类名,count(*)AS 数量
FROM PRODUCTION.PRODUCT A1,
PRODUCTION.PRODUCT_CATEGORY A2,
PRODUCTION.PRODUCT_SUBCATEGORY A3
WHERE A1.PRODUCT_SUBCATEGORYID=A3.PRODUCT_SUBCATEGORYID
AND  A2.PRODUCT_CATEGORYID=A3.PRODUCT_CATEGORYID
AND  A2.NAME='小说'
GROUP  BY  A1.PRODUCT_SUBCATEGORYID,A3.NAME
ORDER  BY  A1.PRODUCT_SUBCATEGORYID;
```

查询结果如表 4.6.2 所示。

表 4.6.2

子分类编号	子分类名	数量
1	世界名著	1
2	武侠	1
4	四大名著	2

使用 GROUP BY 要注意以下问题：

1.在 GROUP BY 子句中的每一列必须明确地命名属于在 FROM 子句中命名的表的一列。分组列的数据类型不能是多媒体数据类型；

2. 分组列不能为集函数表达式或者在 SELECT 子句中定义的别名；

3. 当分组列值包含空值时，则空值作为一个独立组；

4. 当分组列包含多个列名时，则按照 GROUP BY 子句中列出现的顺序进行分组；

5. GROUP BY 子句中至多可包含 255 个分组列；

10.5.2 HAVING 子句的使用

HAVING 子句是 SELECT 语句的可选项部分，它也定义了一个成组表。HAVING 子句语法如下：

```
<HAVING 子句> ::= HAVING <搜索条件>
<搜索条件>::= <表达式>
```

HAVING 子句定义了一个成组表，其中只含有搜索条件为 TRUE 的那些组，且通常跟随一个 GROUP BY 子句。HAVING 子句与组的关系正如 WHERE 子句与表中行的关系。WHERE 子句用于选择表中满足条件的行，而 HAVING 子句用于选择满足条件的组。

例 统计出同一子类别的产品数量大于 1 的子类别名称，数量，并按数量从小到大的顺序排列。

```
SELECT A2.NAME AS 子分类名, COUNT (*)AS 数量
FROM PRODUCTION.PRODUCT A1,
PRODUCTION.PRODUCT_SUBCATEGORY A2
WHERE A1.PRODUCT_SUBCATEGORYID=A2.PRODUCT_SUBCATEGORYID
GROUP  BY  A2.NAME
```

```
HAVING COUNT(*)>1
ORDER  BY  2;
```

查询结果如表 4.6.9 所示。

表 4.6.9

子分类名	数量
四大名著	2

系统执行此语句时，首先将 PRODUCT 表和 PRODUCT_SUBCATEGORY 表中的各行按相同的 SUBCATEGORYID 作连接，再按子类别名的取值进行分组，相同的子类别名为一组，然后对每一组使用集函数 COUNT(*)，统计该组内产品的数量，如此继续，直到最后一组。再选择产品数量大于 1 的组作为查询结果。

10.6 ORDER BY 子句

ORDER BY 子句可以选择性地出现在<查询表达式>之后，它规定了当行由查询返回时应具有的顺序。ORDER BY 子句的语法如下：

```
<ORDER BY 子句> ::= ORDER [SIBLINGS] BY < order_by_list>
< order_by_list>::= < order_by_list>,]<order_by_item>
<order_by_item>::=  <exp> [ASC | DESC] [NULLS FIRST|LAST]
< exp >::=<无符号整数> | <列说明> | <值表达式>
```

使用说明

1. ORDER BY 子句提供了要排序的项目清单和他们的排序顺序：递增顺序(ASC，默认)或是递减顺序(DESC)。它必须跟随在<查询表达式>之后，因为它是在查询计算得出的最终结果上进行操作的；

2. 排序键可以是任何在查询清单中的列的名称，或者是对最终结果表的列计算的表达式(即使这一列不在选择清单中)，也可以是子查询。对于 UNION 查询语句，排序键必须在第一个查询子句中出现；对于 GROUP BY 分组的排序，排序键可以使用集函数，但 GROUP BY 分组中必须包含查询列中所有列；

3. <无符号整数> 对应 SELECT 后结果列的序号。当用<无符号整数>代替列名时，<无符号整数>不应大于 SELECT 后结果列的个数。如下面例子中 ORDER BY 3，因查询结果列只有 2 列，无法进行排序，系统将会报错。若采用其他常量表达式(如：-1，3×6)作为排序列，将不影响最终结果表的行输出顺序；

4. SIBLINGS 关键字必须与 CONNECT BY 一起配合使用，专门用于指定层次查询中相同层次数据返回的顺序。详见 4.12.5 层次查询层内排序；

5. 无论采用何种方式标识想要排序的结果列，它们都不支持多媒体数据类型(如 IMAGE、TEXT、BLOB 和 CLOB)；

6. 当排序列值包含 NULL 时，根据指定的"NULLS FIRST|LAST"决定包含空值的行是排在最前还是最后，缺省为 NULLS FIRST；

7. 当排序列包含多个列名时，系统则按列名从左到右排列的顺序，先按左边列将查询结果排序，当左边排序列值相等时，再按右边排序列排序……如此右推，逐个检查调整，最后得到排序结果；

8. 由于 ORDER BY 只能在最终结果上操作，不能将其放在查询中；

9. 如果 ORDER BY 后面使用集函数，则必须使用 GROUP BY 分组，且 GROUP BY 分组中必须包含查询列中所有列；

10. ORDER BY 子句中至多可包含 255 个排序列。

例 将 RESOURCES.DEPARTMENT 表中的资产总值按从大到小的顺序排列。

```
SELECT * FROM RESOURCES.DEPARTMENT ORDER BY DEPARTMENTID DESC;
```

等价于：

```
SELECT * FROM RESOURCES.DEPARTMENT ORDER BY 1 DESC;
```

查询结果如表 4.7.1 所示。

表 4.7.1

DEPARTMENTID	NAME
5	广告部
4	行政部门
3	人力资源
2	销售部门
1	采购部门

例

```
SELECT * FROM RESOURCES.DEPARTMENT ORDER BY 3;
```

系统报错：无效的 ORDER BY 语句。

第 11 章 数据的插入、删除和修改

DM_SQL 语言的数据更新语句包括：数据插入、数据修改和数据删除三种语句，其中数据插入和修改两种语句使用的格式要求比较严格。在使用时要求对相应基表的定义，如列的个数、各列的排列顺序、数据类型及关键约束、唯一性约束、引用约束、检查约束的内容均要了解得很清楚，否则就很容易出错。下面将分别对这三种语句进行讨论。在讨论中，如不特别说明，各例均使用示例库 BOOKSHOP，用户均为建表者 SYSDBA。

11.1 数据插入语句

数据插入语句用于向已定义好的表中插入单个或成批的数据。

INSERT 语句有两种形式。一种形式是值插入，即构造一行或者多行，并将它们插入到表中；另一种形式为查询插入，它通过<查询表达式>返回一个查询结果集以构造要插入表的一行或多行。

数据插入语句的语法格式如下：

语法格式

```
<插入表达式>::=
INSERT <single_insert_stmt> | <multi_insert_stmt>;

<single_insert_stmt>::=[INTO] <full_tv_name> [<t_alias>] <insert_tail> [<return_into_obj>]
<full_tv_name>::=
    | <单表引用>
```

```
        | [<模式名>.]<基表名>  INDEX <索引名>
        | [<模式名>.]<基表名>  PARTITION (<分区名>)
    | <子查询表达式>
    <单表引用>::=[<模式名>.]<基表或视图名>
    <基表或视图名>::=<基表名>|<视图名>
    <子查询表达式>::=(<查询表达式>) [[AS] <表别名>]
    <t_alias>::=[AS] <表别名>
    <insert_tail>::= [(<列名>{,<列名>})]<insert_action>
    <insert_action>::= VALUES <ins_value>
        | <查询表达式>|(<查询表达式>)
        | (<select_clause>)
        | DEFAULT VALUES
        | TABLE <full_tv_name>
    <select_clause> 数据查询语句

    <multi_insert_stmt>::=ALL <multi_insert_into_list> <查询表达式>
            |[ALL|FIRST]<multi_insert_into_condition_list>
[<multi_insert_into_else>]<查询表达式>
    <multi_insert_into_list>::= <insert_into_single>{<insert_into_single>}
    <insert_into_single>::=
        INTO <full_tv_name> [<t_alias>] [(<列名>{,<列名>})][VALUES <ins_value>]
    <ins_value>::=
     (<expr|DEFAULT>{,<expr|DEFAULT>}){,(<expr|DEFAULT>{,<expr| DEFAULT>})}
    <multi_insert_into_condition_list> ::=
    <insert_into_single_condition>{,< insert_into_single_condition>}
    <INSERT_INTO_SINGLE_CONDITION>::=
     WHEN <BOOL_EXP> THEN <MULTI_INSERT_INTO_LIST>
    <MULTI_INSERT_INTO_ELSE>::= ELSE <MULTI_INSERT_INTO_LIST>
```

参数

1． <模式名>　　指明该表或视图所属的模式，缺省为当前模式；

2． <基表名>　　指明被插入数据的基表的名称；

3． <视图名>　　指明被插入数据的视图的名称，实际上 DM 将数据插入到视图引用的基表中；

4． <列名>　　　表或视图的列的名称。在插入的记录中，这个列表中的每一列都被 VALUES 子句或查询说明赋一个值。如果在此列表中省略了表的一个列名，则 DM 用先前定义好的缺省值插入到这一列中。如果此列表被省略，则在 VALUES 子句和查询中必须为表中的所有列指定值；

5． <INS_VALUE>　　指明在列表中对应的列的插入的列值，如果列表被省略了，插入的列值按照基表中列的定义顺序排列。所有的插入值和系统内部相关存储信息一起构成了一条记录，一条记录的长度不能大于页面大小的一半；

6． <查询表达式>　将一个 SELECT 语句所返回的记录插入表或视图的基表中，子查询中选择的列表必须和 INSERT 语句中列名清单中的列具有相同的数量；带有<查询表达式>的插入方式，称查询插入。插入中使用的<查询表达式>也称为查询说明。

使用说明

1．<基表名>或<视图名>后所跟的<列名>必须是该表中的列，且同一列名不允许出现两次，但排列顺序可以与定义时的顺序不一致；

2．<ins_value>中插入值的个数、类型和顺序要与<列名>一一对应；

3．插入在指定值的时候，可以同时指定多行值，这种叫做多行插入或者批量插入。多行插入不支持列存储表；

4．如果某一<列名>未在 INTO 子句后面出现，则新插入的行在这些列上将取空值或缺省值，如该列在基表定义时说明为 NOT NULL 时将会出错；

5．如果<基表名>或<视图名>后没指定任何<列名>，则隐含指定该表或视图的所有列，这时，新插入的行必须在每个列上均有<插入值>；

6．当使用<子查询表达式>作为 INSERT 的目标时，实际上是对查询表达式的基表进行操作，查询表达式的查询项必须都来源于同一个基表且不能是计算列，查询项所属的基表即是查询表达式的基表，如果查询表达式是带有连接的查询，那么对于连接中视图基表以外的表，连接列上必须是主键或者带有 UNIQUE 约束。不支持 PIVOT/UNPIVOT，不支持 UNION/UNION ALL 查询；

举例说明

例 在 VENDOR 表中插入一条供应商信息：账户号为 00，名称为华中科技大学出版社，活动标志为 1，URL 为空，信誉为 2。

```
INSERT INTO PURCHASING.VENDOR(ACCOUNTNO,NAME,ACTIVEFLAG,WEBURL,CREDIT)
VALUES ('00', '华中科技大学出版社', 1, '', 2);
```

如果需要同时多行插入，则可以用如下的 SQL 语句实现：

```
INSERT INTO PURCHASING.VENDOR(ACCOUNTNO,NAME,ACTIVEFLAG,WEBURL,CREDIT)
VALUES ('00', '华中科技大学出版社', 1, '', 2), ('00', '清华大学出版社', 1, '', 3);
```

在定义 VENDOR 表时，设定了检查约束：CHECK(CREDIT IN(1,2,3,4,5))，说明 CREDIT 只能是 1，2，3，4，5。在插入新数据或修改供应商的 CREDIT 时，系统按检查约束进行检查，如果不满足条件，系统将会报错，多行插入中，每一行都会做检查。

由于 DM 支持标量子查询，标量子查询允许用在标量值合法的地方，因此在数据插入语句的<插入值>位置允许出现标量子查询。

例 将书名为长征的图书的出版社插入到 VENDOR 表中。

```
INSERT INTO PURCHASING.VENDOR(ACCOUNTNO,NAME,ACTIVEFLAG,WEBURL,CREDIT)
VALUES('00',
(SELECT PUBLISHER FROM PRODUCTION.PRODUCT WHERE NAME ='长征'),1,'',1);
```

若是需要插入一批数据时，可使用带<查询说明>的插入语句，如下例所示。

例 构造一个新的基表，表名为 PRODUCT_SELL，用来显示出售的商品名称和购买用户名称，并将查询的数据插入此表中。

```
CREATE TABLE PRODUCTION.PRODUCT_SELL
( PRODUCTNAME VARCHAR(50) NOT NULL,
  CUSTOMERNAME VARCHAR(50) NOT NULL);
INSERT INTO PRODUCTION.PRODUCT_SELL
SELECT DISTINCT T1.NAME , T5.NAME
FROM PRODUCTION.PRODUCT T1, SALES.SALESORDER_DETAIL T2,
       SALES.SALESORDER_HEADER T3, SALES.CUSTOMER T4,
       PERSON.PERSON T5
```

```
    WHERE  T1.PRODUCTID = T2.PRODUCTID  AND  T2.SALESORDERID = T3.SALESORDERID
           AND  T3.CUSTOMERID = T4.CUSTOMERID  AND  T4.PERSONID = T5.PERSONID;
```

该插入语句将已销售的商品名称和购买该商品的用户名称插入到新建的 PRODUCT_SELL 表中。查询结果如下表 5.1.1 所示。

表 5.1.1

PRODUCTNAME	CUSTOMERNAME
红楼梦	刘青
老人与海	刘青

11.2 数据修改语句

数据修改语句用于修改表中已存在的数据。

语法格式

```
UPDATE <单表引用>|<子查询> {<单列修改子句>|<多列修改子句>}
<子查询>::=[<普通表 2>]
<普通表 2>::= (<查询表达式>) [[AS] <表别名> [<新生列>]]
<单列修改子句>::= SET<列名>=<<值表达式>|DEFAULT>{,<列名>=<<值表达式>|DEFAULT>}[FROM <表引用>{,<表引用>}][WHERE <条件表达式>];
<多列修改子句>::= SET <列名>{,<列名>}= <SUBQUERY>;
<单表引用>::= [<模式名>.]<基表或视图名>
<基表或视图名>::= <基表名>|<视图名>
<结果对象>::= <数组>|<变量>
```

参数

1. <模式名> 指明该表或视图所属的模式，缺省为当前用户的缺省模式；
2. <基表名> 指明被修改数据的基表的名称；
3. <视图名> 指明被修改数据的视图的名称，实际上 DM 对视图的基表更新数据；
4. <列名> 表或视图中被更新列的名称，如果 SET 子句中省略列的名称，列的值保持不变；
5. <值表达式> 指明赋予相应列的新值；
6. <条件表达式>指明限制被更新的行必须符合指定的条件，如果省略此子句，则修改表或视图中所有的行。

使用说明

1. SET 后的<列名>不能重复出现；
2. WHERE 子句也可以包含子查询。如果省略了 WHERE 子句，则表示要修改所有的元组；
3. 如果<列名>为被引用列，只有被引用列中未被引用列引用的数据才能被修改；如果<列名>为引用列，引用列的数据被修改后也必须满足引用完整性。在 DM 系统中，以上引用完整性由系统自动检查；
4. 如果视图的定义查询中含有以下结构则不能更新视图：
   1) 联结运算；
   2) 集合运算符；
   3) GROUP BY 子句；

4） 集函数；
5） INTO 语句；
6） HAVING 字句；

举例说明

例 将出版社为中华书局的图书的现在销售价格增加 1 元。

```
UPDATE  PRODUCTION.PRODUCT  SET  NOWPRICE = NOWPRICE + 1.0000
WHERE  PUBLISHER = '中华书局';
```

11.3 数据删除语句

数据删除语句用于删除表中已存在的数据。

语法格式

```
DELETE  [FROM]  <表引用>
[WHERE <条件表达式>][RETURN <列名>{,<列名>} INTO <结果对象>,{<结果对象>}];
<表引用>::= [<模式名>.]<基表或视图名> |
            <子查询表达式>
<基表或视图名>::= <基表名>|<视图名>
<子查询表达式>::=(<查询表达式>) [[AS] <表别名> [<新生列>]]
<结果对象>::=<数组>|<变量>
```

参数

1. <模式名> 指明该表或视图所属的模式，缺省为当前模式；
2. <基表名> 指明被删除数据的基表的名称；
3. <视图名> 指明被删除数据的视图的名称，实际上 DM 将从视图的基表中删除数据；
4. <条件表达式> 指明基表或视图的基表中被删除的记录须满足的条件。

使用说明

1． 如果不带 WHERE 子句，表示删除表中全部元组，但表的定义仍在字典中。因此，DELETE 语句删除的是表中的数据，并未删除表结构；

2． 由于 DELETE 语句一次只能对一个表进行删除，因此当两个表存在引用与被引用关系时，要先删除引用表里的记录，只有引用表中无记录时，才能删被引用表中的记录，否则系统会报错；

3． 如果视图的定义查询中包含以下结构之一，就不能从视图中删除记录：

1） 联结运算；
2） 集合运算符；
3） GROUP BY 子句；
4） 集函数；
5） INTO 语句；
6） HAVING 语句；

4． 当<子查询表达式>作为 DELETE 的目标时，实际上是对查询表达式的基表进行操作，查询表达式的查询项必须都来源于同一个基表且不能是计算列，查询项所属的基表即是查询表达式的基表，如果查询表达式是带有连接的查询，那么对于连接中视图基表以外的表，连接列上必须是主键或者带有 UNIQUE 约束。不支持 PIVOT/UNPIVOT，不支持 UNION/UNION ALL 查询；

例 将没有分配部门的员工的住址信息删除。

```
DELETE  FROM  RESOURCES.EMPLOYEE_ADDRESS
WHERE  EMPLOYEEID  IN
( SELECT  EMPLOYEEID
FROM  RESOURCES.EMPLOYEE
WHERE  EMPLOYEEID  NOT  IN
( SELECT  EMPLOYEEID  FROM  RESOURCES.EMPLOYEE_DEPARTMENT));
```

第 12 章 自主访问控制

自主访问控制（Discretionary Access Control，DAC）是这样一种访问控制方式：由数据库对象的拥有者自主决定是否将自己拥有的对象的部分或全部访问权限授予其他用户。也就是说，在自主访问控制下，用户可以按照自己的意愿，有选择地与其他用户共享他拥有的数据库对象。

12.1 权限管理

DM 数据库对用户的权限管理有着严密的规定，如果没有权限，用户将无法完成任何操作。

用户权限有两类：数据库权限和对象权限。数据库权限主要是指针对数据库对象的创建、删除、修改的权限，对数据库备份等权限。而对象权限主要是指对数据库对象中的数据的访问权限。数据库权限一般由 SYSDBA、SYSAUDITOR 和 SYSSSO 指定，也可以由具有特权的其他用户授予。对象权限一般由数据库对象的所有者授予用户，也可由 SYSDBA 用户指定，或者由具有该对象权限的其他用户授权。

12.1.1 数据库权限

数据库权限是与数据库安全相关的非常重要的权限，其权限范围比对象权限更加广泛，因而一般被授予数据库管理员或者一些具有管理功能的角色。数据库权限与 DM 预定义角色有着重要的联系，一些数据库权限由于权力较大，只集中在几个 DM 系统预定义角色中，且不能转授，我们将在 3.2.1.1 中进行具体介绍。

DM 提供了 100 余种数据库权限，表 3.1 列出了最常用的几种数据库权限。完整的数据库权限列表将在 3.2.1.1 中与 DM 预定义角色一起进行介绍。

表 3.1 常用的几种数据库权限

数据库权限	说明
CREATE TABLE	在自己的模式中创建表的权限
CREATE VIEW	在自己的模式中创建视图的权限
CREATE USER	创建用户的权限
CREATE TRIGGER	在自己的模式中创建触发器的权限
ALTER USER	修改用户的权限

ALTER DATABASE	修改数据库的权限
CREATE PROCEDURE	在自己模式中创建存储程序的权限

不同类型的数据库对象，其相关的数据库权限也不相同。

例如，对于表对象，相关的数据库权限包括：

- CREATE TABLE：创建表
- CREATE ANY TABLE：在任意模式下创建表
- ALTER ANY TABLE：修改任意表
- DROP ANY TABLE：删除任意表
- INSERT TABLE：插入表记录
- INSERT ANY TABLE：向任意表插入记录
- UPDATE TABLE：更新表记录
- UPDATE ANY TABLE：更新任意表的记录
- DELETE TABLE：删除表记录
- DELETE ANY TABLE：删除任意表的记录
- SELECT TABLE：查询表记录
- SELECT ANY TABLE：查询任意表的记录
- REFERENCES TABLE：引用表
- REFERENCES ANY TABLE：引用任意表
- DUMP TABLE：导出表
- DUMP ANY TABLE：导出任意表
- GRANT TABLE：向其他用户进行表上权限的授权
- GRANT ANY TABLE：向其他用户进行任意表上权限的授权

而对于存储程序对象，其相关的数据库权限则包括：

- CREATE PROCEDURE：创建存储程序
- CREATE ANY PROCEDURE：在任意模式下创建存储程序
- DROP PROCEDURE：删除存储程序
- DROP ANY PROCEDURE：删除任意存储程序
- EXECUTE PROCEDURE：执行存储程序
- EXECUTE ANY PROCEDURE：执行任意存储程序
- GRANT PROCEDURE：向其他用户进行存储程序上权限的授权
- GRANT ANY PROCEDURE：向其他用户进行任意存储程序上权限的授权

需要说明的是，表、视图、触发器、存储程序等对象为模式对象，在默认情况下对这些对象的操作都是在当前用户自己的模式下进行的。如果要在其他用户的模式下操作这些类型的对象，需要具有相应的 ANY 权限。例如，要能够在其他用户的模式下创建表，当前用户必须具有 CREATE ANY TABLE 数据库权限，如果希望能够在其他用户的模式下删除表，必须具有 DROP ANY TABLE 数据库权限。

12.1.2 对象权限

对象权限主要是对数据库对象中的数据的访问权限，主要用来授予需要对某个数据库对象的数据进行操纵的数据库普通用户。表 3.2 列出了主要的对象权限。

表 3.2 常用的对象权限

数据库对象类型 对象权限	表	视图	存储程序	包	类	类型	序列	目录	域
SELECT	√	√					√		
INSERT	√	√							
DELETE	√	√							
UPDATE	√	√							
REFERENCES	√								
DUMP	√								
EXECUTE			√	√	√	√		√	
READ								√	
WRITE								√	
USAGE									√

SELECT、INSERT、DELETE 和 UPDATE 权限分别是针对数据库对象中的数据的查询、插入、删除和修改的权限。对于表和视图来说，删除操作是整行进行的，而查询、插入和修改却可以在一行的某个列上进行，所以在指定权限时，DELETE 权限只要指定所要访问的表就可以了，而 SELECT、INSERT 和 UPDATE 权限还可以进一步指定是对哪个列的权限。

表对象的 REFERENCES 权限是指可以与一个表建立关联关系的权限，如果具有了这个权限，当前用户就可以通过自己的一个表中的外键，与对方的表建立关联。关联关系是通过主键和外键进行的，所以在授予这个权限时，可以指定表中的列，也可以不指定。

存储程序等对象的 EXECUTE 权限是指可以执行这些对象的权限。有了这个权限，一个用户就可以执行另一个用户的存储程序、包、类等。

目录对象的 READ 和 WRITE 权限指可以读或写访问某个目录对象的权限。

域对象的 USAGE 权限指可以使用某个域对象的权限。拥有某个域的 USAGE 权限的用户可以在定义或修改表时为表列声明使用这个域。

当一个用户获得另一个用户的某个对象的访问权限后，可以以"模式名.对象名"的形式访问这个数据库对象。一个用户所拥有的对象和可以访问的对象是不同的，这一点在数据字典视图中有所反映。在默认情况下用户可以直接访问自己模式中的数据库对象，但是要访问其他用户所拥有的对象，就必须具有相应的对象权限。

对象权限的授予一般由对象的所有者完成，也可由 SYSDBA 或具有某对象权限且具有转授权限的用户授予，但最好由对象的所有者完成。

12.2 角色管理

角色是一组权限的组合，使用角色的目的是使权限管理更加方便。假设有 10 个用户，这些用户为了访问数据库，至少拥有 CREATE TABLE、CREATE VIEW 等权限。如果将这些权限分别授予这些用户，那么需要进行的授权次数是比较多的。但是如果把这些权限事先放在一起，然后作为一个整体授予这些用户，那么每个用户只需一次授权，授权的次数将大大减少，而且用户数越多，需要指定的权限越多，这种授权方式的优越性就越明显。这些事先组合在一起的一组权限就是角色，角色中的权限既可以是数据库权限，也可以是对象权限，还可以是别的角色。

为了使用角色，首先在数据库中创建一个角色，这时角色中没有任何权限。然后向角色中添加权限。最后将这个角色授予用户，这个用户就具有了角色中的所有权限。在使用角色的过程中，可以随时向角色中添加权限，也可以随时从角色中删除权限，用户的权限也随之改变。如果要回收所有权限，只需将角色从用户回收即可。

12.2.1 角色的创建与删除

1) 3.2.1.1 DM 预定义角色

在 DM 数据库中有两类角色，一类是 DM 预设定的角色，一类是用户自定义的角色。DM 提供了一系列的预定义角色以帮助用户进行数据库权限的管理。预定义角色在数据库被创建之后即存在，并且已经包含了一些权限，数据库管理员可以将这些角色直接授予用户。

在"三权分立"和"四权分立"机制下，DM 的预定义角色及其所具有的权限是不相同的。表 3.3 和表 3.4 分别列出了"三权分立"和"四权分立"机制下常见的系统角色及其简单介绍，完整的预定义角色权限列表请参见附录 1 和附录 2。

表 3.3 "三权分立"常见的数据库预设定的角色

角色名称	角色简单说明
DBA	DM 数据库系统中对象与数据操作的最高权限集合，拥有构建数据库的全部特权，只有 DBA 才可以创建数据库结构
RESOURCE	可以创建数据库对象，对有权限的数据库对象进行数据操纵，不可以创建数据库结构
PUBLIC	不可以创建数据库对象，只能对有权限的数据库对象进行数据操纵
VTI	具有系统动态视图的查询权限，VTI 默认授权给 DBA 且可转授
SOI	具有系统表的查询权限
DB_AUDIT_ADMIN	数据库审计的最高权限集合，可以对数据库进行各种审计操作，并创建新的审计用户
DB_AUDIT_OPER	可以对数据库进行各种审计操作，但不能创建新的审计用户
DB_AUDIT_PUBLIC	不能进行审计设置，但可以查询审计相关字典表
DB_AUDIT_VTI	具有系统动态视图的查询权限，DB_AUDIT_VTI 默认授权给 DB_AUDIT_ADMIN 且可转授
DB_AUDIT_SOI	具有系统表的查询权限
DB_POLICY_ADMIN	数据库强制访问控制的最高权限集合，可以对数据库进行强制访问控制管理，并创建新的安全管理用户
DB_POLICY_OPER	可以对数据库进行强制访问控制管理，但不能创建新的安全管理用户
DB_POLICY_PUBLIC	不能进行强制访问控制管理，但可以查询强制访问控制相关字典表
DB_POLICY_VTI	具有系统动态视图的查询权限，DB_POLICY_VTI 默认授权给 DB_POLICY_ADMIN 且可转授
DB_POLICY_SOI	具有系统表的查询权限

达梦数据库信息资料

表 3.4 “四权分立”常见的数据库预设定的角色

角色名称	角色简单说明
DBA	拥有构建数据库的全部特权，只有 DBA 才可以创建数据库结构
RESOURCE	可以创建和删除角色
PUBLIC	只能查询相关的数据字典表
VTI	具有系统动态视图的查询权限，VTI 默认授权给 DBA 且可转授
SOI	具有系统表的查询权限
DB_AUDIT_ADMIN	数据库审计的最高权限集合，可以对数据库进行各种审计操作，并创建新的审计用户
DB_AUDIT_OPER	可以对数据库进行各种审计操作，但不能创建新的审计用户
DB_AUDIT_PUBLIC	不能进行审计设置，但可以查询审计相关字典表
DB_AUDIT_VTI	具有系统动态视图的查询权限，DB_AUDIT_VTI 默认授权给 DB_AUDIT_ADMIN 且可转授
DB_AUDIT_SOI	具有系统表的查询权限
DB_POLICY_ADMIN	数据库强制访问控制的最高权限集合，可以对数据库进行强制访问控制管理，并创建新的安全管理用户
DB_POLICY_OPER	可以对数据库进行强制访问控制管理，但不能创建新的安全管理用户
DB_POLICY_PUBLIC	不能进行强制访问控制管理，但可以查询强制访问控制相关字典表
DB_POLICY_VTI	具有系统动态视图的查询权限，DB_POLICY_VTI 默认授权给 DB_POLICY_ADMIN 且可转授
DB_POLICY_SOI	具有系统表的查询权限
DB_OBJECT_ADMIN	可以在自己的模式下创建各种数据库对象并进行数据操纵，也可以创建和删除非模式对象
DB_OBJECT_OPER	可以在自己的模式下创建数据库对象并进行数据操纵
DB_OBJECT_PUBLIC	不可以创建数据库对象，只能对有权限的数据库对象进行数据操纵
DB_OBJECT_VTI	具有系统动态视图的查询权限，DB_OBJECT_VTI 默认授权给 DB_OBJECT_ADMIN 且可转授
DB_OBJECT_SOI	具有系统表的查询权限

我们可以认为“三权分立”安全机制将用户分为了三种类型，“四权分立”安全机制将用户分为了四种类型，每种类型又各对应四种预定义角色。如：

- DBA 类型对应 DBA、RESOURCE、PUBLIC、VTI、SOI 预定义角色；
- AUDITOR 对应 DB_ADUTI_ADMIN、DB_AUDIT_OPER、DB_AUDIT_PUBLIC、DB_AUDIT_VTI、DB_AUDIT_SOI 预定义角色；
- SSO 对应 DB_POLICY_ADMIN、DB_POLICY_OPER、DB_POLICY_PUBLIC、

达梦数据库信创大赛资料

DB_POLICY_VTI、DB_POLICY_SOI 预定义角色；
- DBO 对应 DB_OBJECT_ADMIN、DB_OBJECT_OPER、DB_OBJECT_PUBLIC、DB_OBJECT_VTI、DB_OBJECT_SOI 预定义角色。

初始时仅有管理员具有创建用户的权限，每种类型的管理员创建的用户缺省就拥有这种类型的 PUBLIC 和 SOI 预定义角色，如 SYSAUDITOR 新创建的用户缺省就具有 DB_AUDIT_PUBLIC 和 DB_AUDIT_SOI 角色。之后管理员可根据需要进一步授予新建用户其他预定义角色。

管理员也可以将"CREATE USER"权限转授给其他用户，这些用户之后就可以创建新的用户了，他们创建的新用户缺省也具有与其创建者相同类型的 PUBLIC 和 SOI 预定义角色。

■ **限制 DBA 的"ANY"权限**

从附录 1 和附录 2 的表中我们可以看到，缺省情况下，DBA 角色拥有许多的"ANY"权限，如 DROP ANY TABLE、INSERT ANY TABLE 等等。在 DM 安全版本下，提供了一种限制 DBA 的"ANY"权限的方法。只有具有 DB_POLICY_ADMIN 角色的用户才能限制 DBA 的"ANY"权限，通过执行下面的系统过程来实现：

```
VOID
SP_RESTRICT_DBA(
    FLAG        INT
);
```

参数 FLAG 置为 1 表示限制 DBA 的"ANY"权限，此过程执行后所有的 DBA 用户将不再具有"ANY"权限；FLAG 置为 0 表示不限制 DBA 的"ANY"权限。

2)　3.2.1.2 创建角色

具有"CREATE ROLE"数据库权限的用户也可以创建新的角色，其语法如下：

```
CREATE ROLE  <角色名>;
```

使用说明：

1. 创建者必须具有 CREATE ROLE 数据库权限；
2. 角色名的长度不能超过 128 个字符；
3. 角色名不允许和系统已存在的用户名重名；
4. 角色名不允许是 DM 保留字。

例如，创建角色 BOOKSHOP_ROLE1，赋予其 PERSON.ADDRESS 表的 SELECT 权限。

```
CREATE ROLE BOOKSHOP_ROLE1;
GRANT SELECT ON PERSON.ADDRESS TO BOOKSHOP_ROLE1;
```

3)　3.2.1.3 删除角色

具有"DROP ROLE"权限的用户可以删除角色，其语法如下：

```
DROP ROLE  <角色名>;
```

即使已将角色授予了其他用户，删除这个角色的操作也将成功。此时，那些之前被授予

该角色的用户将不再具有这个角色所拥有的权限，除非用户通过其他途径也获得了这个角色所具有的权限。

例如，接上一小节的例子。将角色 BOOKSHOP_ROLE1 授予用户 BOOKSHOP_USER，则用户 BOOKSHOP_USER 具有了 SELECT 表 PERSON.ADDRESS 的权限。此时删除角色 BOOKSHOP_ROLE1，那么用户 BOOKSHOP_USER 将不再能查询表 PERSON.ADDRESS。但是如果用户 BOOKSHOP_USER 从别的途径（直接授权或通过别的角色授权）重复获得了 SELECT 表 PERSON.ADDRESS 的权限，那么即使删除了角色 BOOKSHOP_ROLE1，用户 BOOKSHOP_USER 仍然具有 SELECT 表 PERSON.ADDRESS 的权限。

12.2.2 角色的启用与禁用

某些时候，用户不愿意删除一个角色，但是却希望这个角色失效，此时可以使用 DM 系统过程 SP_SET_ROLE 来设置这个角色为不可用，将第二参数置为 0 表示禁用角色。

例如，下面的语句将角色 BOOKSHOP_ROLE1 禁用。

```
SP_SET_ROLE('BOOKSHOP_ROLE1', 0);
```

使用说明：

1. 只有拥有 ADMIN_ANY_ROLE 权限的用户才能启用和禁用角色，并且设置后立即生效；
2. 凡是包含禁用角色 A 的角色 M，M 中禁用的角色 A 将无效，但是 M 仍有效；
3. 系统预设的角色是不能设置的，如：DBA、PUBLIC、RESOURCE。

当用户希望启用某个角色时，同样可以通过 SP_SET_ROLE 来启用角色，只要将第二个参数置为 1 即可。

例如，下面的语句将启用角色 BOOKSHOP_ROLE1。

```
SP_SET_ROLE('BOOKSHOP_ROLE1', 1);
```

12.3 权限的分配与回收

可以通过 GRANT 语句将权限（包括数据库权限、对象权限以及角色权限）分配给用户和角色，之后也可以使用 REVOKE 语句将授出的权限再进行回收。

12.3.1 数据库权限的分配与回收

1) 3.3.1.1 数据库权限的分配

可以使用 GRANT 语句授予用户和角色数据库权限。

数据库权限的授权语句语法为：

```
GRANT <特权> TO <用户或角色>{,<用户或角色>} [WITH  ADMIN OPTION];
<特权> ::= <数据库权限>{,<数据库权限>};
<用户或角色>::= <用户名> | <角色名>
```

使用说明：

1. 授权者必须具有对应的数据库权限以及其转授权；
2. 接受者必须与授权者用户类型一致；
3. 如果有 WITH ADMIN OPTION 选项，接受者可以再把这些权限转授给其他用户/角色。

例如，系统管理员 SYSDBA 把建表和建视图的权限授给用户 BOOKSHOP_USER1，并允许其转授。

```
GRANT CREATE TABLE, CREATE VIEW TO BOOKSHOP_USER1 WITH ADMIN OPTION;
```

2) 3.3.1.2 数据库权限的回收

可以使用 REVOKE 语句回收授出的指定数据库权限。

回收数据库权限的语句语法为：

```
REVOKE [ADMIN OPTION FOR]<特权> FROM <用户或角色>{,<用户或角色>} ;
<特权> ::= <数据库权限>{,<数据库权限>}
<用户或角色>::= <用户名> | <角色名>
```

使用说明：

1. 权限回收者必须是具有回收相应数据库权限以及转授权的用户；
2. ADMIN OPTION FOR 选项的意义是取消用户或角色的转授权限，但是权限不回收。

例 1 接 3.3.1.1 中的例子，现在 SYSDBA 把用户 BOOKSHOP_USER1 的建表权限收回。

```
REVOKE CREATE TABLE FROM BOOKSHOP_USER1;
```

例 2 接 3.3.1.1 中的例子，之前 SYSDBA 把建视图的权限授给了用户 BOOKSHOP_USER1，并且允许转授。现在 SYSDBA 不让用户 BOOKSHOP_USER1 转授 CREATE VIEW 权限。

```
REVOKE ADMIN OPTION FOR CREATE VIEW FROM BOOKSHOP_USER1;
```

BOOKSHOP_USER1 仍有 CREATE VIEW 权限，但是不能将 CREATE VIEW 权限转授给其他用户。

3) 3.3.1.3 限制相关数据库权限的授予与回收

通过 INI 参数 ENABLE_DDL_ANY_PRIV 限制 DDL 相关的 ANY 数据库权限的授予与回收，有 2 个取值：

- 1：可以授予和回收 DDL 相关的 ANY 系统权限；
- 0：不可以授予和回收 DDL 相关的 ANY 系统权限。缺省为 0。

例 1 当 ENABLE_DDL_ANY_PRIV=0 时，禁止授予或回收 create any trigger 的权限。

```
CONN SYSDBA/SYSDBA
create user DBSEC  identified by 123456789;
GRANT create any trigger TO DBSEC;     --报错
revoke create any trigger from DBSEC;--报错
```

12.3.2 对象权限的分配与回收

1) 3.3.2.1 对象权限的分配

可以使用 GRANT 语句将对象权限授予用户和角色。

对象权限的授权语句语法为：

```
GRANT <特权> ON [<对象类型>] <对象> TO <用户或角色>{,<用户或角色>} [WITH GRANT OPTION];
<特权>::= ALL [PRIVILEGES] | <动作> {, <动作>}
<动作>::= SELECT[(<列清单>)] |
          INSERT[(<列清单>)] |
          UPDATE[(<列清单>)] |
          DELETE |
          REFERENCES[(<列清单>)] |
          EXECUTE|
          READ|
          WRITE|
          USAGE
<列清单>::= <列名> {,<列名>}
<对象类型>::= TABLE | VIEW | PROCEDURE | PACKAGE | CLASS | TYPE | SEQUENCE | DIRECTORY
| DOMAIN
<对象> ::= [<模式名>.]<对象名>
<对象名> ::= <表名> | <视图名> | <存储过程/函数名> |<包名> |<类名> |<类型名> |<序列名>
| <目录名> | <域名>
<用户或角色>::= <用户名> | <角色名>
```

使用说明：

1. 授权者必须是具有对应对象权限以及其转授权的用户；
2. 如未指定对象的<模式名>，模式为授权者所在的模式。DIRECTORY 为非模式对象，没有模式；
3. 如设定了对象类型，则该类型必须与对象的实际类型一致，否则会报错；
4. 带 WITH GRANT OPTION 授予权限给用户时，则接受权限的用户可转授此权限；
5. 不带列清单授权时，如果对象上存在同类型的列权限，会全部自动合并；
6. 对于用户所在的模式的表，用户具有所有权限而不需特别指定；

当授权语句中使用了 ALL PRIVILEGES 时，会将指定的数据库对象上所有的对象权限都授予被授权者。不同类型的数据库对象相关的对象权限见 3.1.2 节的介绍。

例 1 SYSDBA 把 PERSON.ADDRESS 表的全部权限授给用户 BOOKSHOP_USER1。

```
GRANT SELECT, INSERT, DELETE, UPDATE, REFERENCES ON PERSON.ADDRESS TO
BOOKSHOP_USER1;
```

该语句也可写为以下形式：

```
GRANT ALL PRIVILEGES ON PERSON.ADDRESS TO BOOKSHOP_USER1;
```

例 2 假设用户 BOOKSHOP_USER1 创建了存储过程 BOOKSHOP_USER1_PROC1，数据库管理员 SYSDBA 把该存储过程的执行权 EXECUTE 授给已存在用户 BOOKSHOP_USER2，

并使其具有该权限的转授权。

```
GRANT EXECUTE ON PROCEDURE BOOKSHOP_USER1.BOOKSHOP_USER1_PROC1
TO BOOKSHOP_USER2 WITH GRANT OPTION;
```

例 3 假设 SYSDBA 是表 BOOKSHOP_T1 的创建者，用户 BOOKSHOP_USER1、BOOKSHOP_USER2、BOOKSHOP_USER3 存在，且都不是 DBA 权限用户。

```
(1)以 SYSDBA 身份登录，并执行语句：
GRANT SELECT ON BOOKSHOP_T1 TO BOOKSHOP_USER1 WITH GRANT OPTION;
/*正确*/
(2)以 BOOKSHOP_USER1 身份登录，并执行语句：
GRANT SELECT ON SYSDBA.BOOKSHOP_T1 TO BOOKSHOP_USER2;
/*正确*/
(3)以 BOOKSHOP_USER2 身份登录，并执行语句：
GRANT SELECT ON SYSDBA.BOOKSHOP_T1 TO BOOKSHOP_USER3;
/*错误，用户 BOOKSHOP_USER2 没有 SELECT 权限的转授权*/
```

例 4 用户 SYSDBA 创建一个名为 V_PRODUCT 的视图，要求对于该视图，允许 BOOKSHOP_USER1 能够进行查询、插入、删除和更新操作，用户 BOOKSHOP_USER2 和 BOOKSHOP_USER3 也可进行同样的工作，但要求其操作权限由用户 BOOKSHOP_USER1 控制。

```
(1)以 SYSDBA 的身份登录：
CREATE VIEW V_PRODUCT AS
SELECT * FROM PRODUCTION.PRODUCT
WHERE NOWPRICE>=20 AND ORIGINALPRICE<40 WITH CHECK OPTION;

GRANT SELECT, INSERT, DELETE, UPDATE ON V_PRODUCT
TO BOOKSHOP_USER1 WITH GRANT OPTION;

(2)以用户 BOOKSHOP_USER1 的身份登录：
GRANT SELECT, INSERT, DELETE, UPDATE ON SYSDBA.V_PRODUCT TO BOOKSHOP_USER2,
BOOKSHOP_USER3;
```

例 5 假定表的创建者 SYSDBA 把所创建的 PRODUCTION.PRODUCT 表的部分列的更新权限授予用户 BOOKSHOP_USER1，BOOKSHOP_USER1 再将此更新权限转授给用户 BOOKSHOP_USER2。

```
(1)以 SYSDBA 的身份登录，并执行语句：
GRANT UPDATE (ORIGINALPRICE, NOWPRICE) ON PRODUCTION.PRODUCT TO BOOKSHOP_USER1
WITH GRANT OPTION;
(2)以 BOOKSHOP_USER1 的身份登录，并执行语句，把列级更新权限授给用户 BOOKSHOP_USER2：
GRANT UPDATE(ORIGINALPRICE, NOWPRICE)ON PRODUCTION.PRODUCTTO BOOKSHOP_USER2;
```

2) 3.3.2.2 对象权限的回收

可以使用 REVOKE 语句回收授出的指定数据库对象的指定权限。

对象权限的回收语句语法为：

```
REVOKE [GRANT OPTION FOR] <特权> ON [<对象类型>]<对象> FROM <用户或角色> {,<用户或
```

```
角色>} [<回收选项>];
<特权>::= ALL [PRIVILEGES] | <动作> {, <动作>}
<动作>::= SELECT |
          INSERT |
          UPDATE |
          DELETE |
          REFERENCES |
          EXECUTE|
          READ|
          WRITE|
          USAGE
<对象类型>::= TABLE | VIEW | PROCEDURE | PACKAGE | CLASS | TYPE | SEQUENCE | DIRECTORY
| DOMAIN
<对象> ::= [<模式名>.]<对象名>
<对象名> ::= <表名> | <视图名> | <存储过程/函数名> |<包名> |<类名> |<类型名> | <序列名>
| <目录名> | <域名>
<用户或角色>::= <用户名> | <角色名>
<回收选项> ::= RESTRICT | CASCADE
```

使用说明：

1. 权限回收者必须是具有回收相应对象权限以及转授权的用户；
2. 回收时不能带列清单，若对象上存在同类型的列权限，则一并被回收；
3. 使用 GRANT OPTION FOR 选项的目的是收回用户或角色权限转授的权利，而不回收用户或角色的权限；并且 GRANT OPTION FOR 选项不能和 RESTRICT 一起使用，否则会报错；
4. 在回收权限时，设定不同的回收选项，其意义不同。具体如下：
   - 若不设定回收选项，无法回收授予时带 WITH GRANT OPTION 的权限，但也不会检查要回收的权限是否存在限制；
   - 若设定为 RESTRICT，无法回收授予时带 WITH GRANT OPTION 的权限，也无法回收存在限制的权限，如角色上的某权限被别的用户用于创建视图等；
   - 若设定为 CASCADE，可回收授予时带或不带 WITH GRANT OPTION 的权限，若带 WITH GRANT OPTION 还会引起级联回收。利用此选项时也不会检查权限是否存在限制。另外，利用此选项进行级联回收时，若被回收对象上存在另一条路径授予同样权限给该对象时，则仅需回收当前权限。

说明：**用户 A 给用户 B 授权且允许其转授，B 将权限转授给 C。当 A 回收 B 的权限的时候必须加 CASCADE 回收选项。**

例 1 接 3.3.2.1 节例 1，SYSDBA 从用户 BOOKSHOP_USER1 处回收其授出的 PERSON.ADDRESS 表的全部权限。

```
REVOKE ALL PRIVILEGES ON BOOKSHOP_T1 FROM BOOKSHOP_USER1 CASCADE;
```

例 2 接 3.3.2.1 节例 2，SYSDBA 从用户 BOOKSHOP_USER2 处回收其授出的存储过程 BOOKSHOP_USER1_PROC1 的 EXECUTE 权限。

```
REVOKE EXECUTE ON PROCEDURE BOOKSHOP_USER1.BOOKSHOP_USER1_PROC1
FROM BOOKSHOP_USER2 CASCADE;
```

例 3 假定系统中存在非 SYSDBA 用户 BOOKSHOP_USER1、BOOKSHOP_USER2、BOOKSHOP_USER3 和 BOOKSHOP_USER4，其中 BOOKSHOP_USER1 具有 CREATE TABLE 数据库权限，BOOKSHOP_USER2 具有 CREATE VIEW 数据库权限，且已成功执行了如下的语句：

```
BOOKSHOP_USER1 登录执行:
CREATE TABLE T1(ID INT, NAME VARCHAR(100));
GRANT SELECT ON T1 TO PUBLIC;
GRANT INSERT(ID) ON T1 TO BOOKSHOP_USER2 WITH GRANT OPTION;
BOOKSHOP_USER2 登录执行:
CREATE VIEW V1 AS SELECT NAME FROM BOOKSHOP_USER1.T1 WHERE ID < 10;
GRANT INSERT(ID) ON BOOKSHOP_USER1.T1 TO BOOKSHOP_USER3 WITH GRANT OPTION;
BOOKSHOP_USER3 登录执行:
GRANT INSERT(ID) ON BOOKSHOP_USER1.T1 TO BOOKSHOP_USER4 WITH GRANT OPTION;
BOOKSHOP_USER4 登录执行:
GRANT INSERT(ID) ON BOOKSHOP_USER1.T1 TO BOOKSHOP_USER2 WITH GRANT OPTION;
```

此时，若 BOOKSHOP_USER1 执行以下语句会引起之前授予的 INSERT(ID) 权限被合并。

```
GRANT INSERT ON T1 TO BOOKSHOP_USER2 WITH GRANT OPTION;
```

若 BOOKSHOP_USER1 执行以下语句，会成功，因为这种方式不检查权限是否存在限制。

```
REVOKE SELECT ON T1 FROM PUBLIC;
```

若 BOOKSHOP_USER1 执行以下语句，则会失败，因为这种方式检查权限是否存在限制，即 BOOKSHOP_USER2 利用此权限创建了视图 V1。

```
REVOKE SELECT ON T1 FROM PUBLIC RESTRICT;
```

若 BOOKSHOP_USER1 执行以下语句，会进行级联回收 BOOKSHOP_USER2、BOOKSHOP_USER3、BOOKSHOP_USER4 授出的 INSERT(ID) 权限。

```
REVOKE INSERT ON T1 FROM BOOKSHOP_USER2 CASCADE;
```

但若 BOOKSHOP_USER1 连续执行了以下两条语句，则后一条语句仅回收其授予 BOOKSHOP_USER2 的权限，而不会产生级联回收，因为有另一条路经授予了 BOOKSHOP_USER3 同样的权限 INSERT(ID)。

```
GRANT INSERT ON T1 TO BOOKSHOP_USER3 WITH GRANT OPTION;
REVOKE INSERT ON T1 FROM BOOKSHOP_USER2 CASCADE;
```

12.3.3 角色权限的分配与回收

1) 3.3.3.1 角色权限的分配

通常角色包含权限或其他角色，通过使用 GRANT 语句将一个角色授予用户或另一角色可以使得用户和角色继承该角色所具有的权限。

授予角色权限的语句语法如下：

```
GRANT <角色名>{, <角色名>} TO <用户或角色>{,<用户或角色>} [WITH  ADMIN OPTION];
<用户名或角色名> ::= <用户名> | <角色名>
```

使用说明：

1. 角色的授予者必须为拥有相应的角色以及其转授权的用户；
2. 接受者必须与授权者类型一致(譬如不能把审计角色授予标记角色)；
3. 支持角色的转授；
4. 不支持角色的循环转授，如将 BOOKSHOP_ROLE1 授予 BOOKSHOP_ROLE2，BOOKSHOP_ROLE2 不能再授予 BOOKSHOP_ROLE1。

例如，假设存在角色 BOOKSHOP_ROLE1，角色 BOOKSHOP_ROLE2，用户 BOOKSHOP_USER1。

```
/*让用户 BOOKSHOP_USER1 继承角色 BOOKSHOP_ROLE1 的权限*/
GRANT BOOKSHOP_ROLE1 TO BOOKSHOP_USER1;
/*让角色 BOOKSHOP_ROLE2 继承角色 BOOKSHOP_ROLE1 的权限*/
GRANT BOOKSHOP_ROLE1 TO BOOKSHOP_ROLE2;
```

2) 3.3.3.2 角色权限的回收

可以使用 REVOKE 语句回收用户或其它角色从指定角色继承过来的权限。

回收角色权限的语句语法为：

```
REVOKE  [ADMIN OPTION FOR] <角色名>{,<角色名>} FROM  <角色名或用户名>;
<角色名或用户名> ::= <用户名> | <角色名>
```

使用说明：

1. 权限回收者必须是具有回收相应角色以及转授权的用户；
2. 使用 GRANT OPTION FOR 选项的目的是收回用户或角色权限转授的权利，而不回收用户或角色的权限。

例如，接 3.3.3.1 节的例子，回收用户 BOOKSHOP_USER1 和角色 BOOKSHOP_ROLE2 的 BOOKSHOP_ROLE1 角色。

```
REVOKE BOOKSHOP_ROLE1 FROM BOOKSHOP_USER1;
REVOKE BOOKSHOP_ROLE1 FROM BOOKSHOP_ROLE2;
```